САНКТ-ПЕТЕРБУРГСКИЙ
ГОСУДАРСТВЕННЫЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И УПРАВЛЕНИЯ

Е. Г. Павловский, В. А. Жвариков, А.А. Кузьмин

ОРГАНИЗАЦИЯ И ОСНОВЫ ПРОГРАММИРОВАНИЯ МИКРОПРОЦЕССОРОВ

Учебное пособие и методические указания
к лабораторному практикуму

Санкт-Петербург
2014

УДК 004.2

Е. Г. Павловский, В. А. Жвариков, А. А. Кузьмин. **Организация и основы программирования микропроцессоров:** Учебное пособие и методические указания к лабораторному практикуму. — СПб.: Санкт-Петербургский государственный политехнический университет, 2014. — 130 с., ил.

В пособии излагаются основные принципы организации и функционирования центрального ядра ЭВМ, построенного на базе микропроцессоров различных классов. Приводится описание архитектуры отдельных микропроцессорных БИС. Рассматриваются особенности блоков управления, используемые способы адресации, системы команд и некоторые вопросы взаимодействия микропроцессоров с памятью и внешними устройствами. Настоящее учебное пособие является новой переработанной редакцией ранее изданного учебного пособия «Основы организации ЭВМ и микропроцессоров», авторы: Е.Г.Павловский, В.А.Жвариков. Теоретический материал, изложенный в пособии, поддержан лабораторным практикумом на базе микропроцессорных тренажеров, позволяющих проводить исследования как микропроцессоров с фиксированной разрядностью и списком команд, так и модульных (секционных) с микропрограммным управлением. Пособие предназначено для студентов, обучающихся по направлениям «Информатика и вычислительная техника» и «Управление в технических системах», и призвано помочь студентам в закреплении знаний по курсам «ЭВМ и периферийные устройства», «Вычислительные машины, системы и сети», «Микропроцессорные системы».

Печатается по решению кафедры компьютерных систем и программных технологий Института информационных технологий и управления Санкт-Петербургского государственного политехнического университета.

СОДЕРЖАНИЕ

Предисловие

В современном мире невозможно представить специалиста любой области знаний, не владеющего компьютером. С помощью персональных компьютеров (ПК) решается бесконечно большое число пользовательских задач, и нет сомнения, что области применения компьютеров будут расширяться. ПК, работающие под управлением современных операционных систем, являются сложными программно-аппаратными комплексами. Многие пользователи, успешно применяющие ПК в исследованиях и работе, в большинстве своем пользуются компьютером как «черным ящиком», не особенно представляя себе архитектуру, состав и организацию вычислительной машины (ВМ) в целом. Для основной массы пользователей такое «потребительское» отношение к ВМ удобно и, скорее всего, является, чуть ли, единственно возможным. Совсем иначе обстоит дело со специалистами в области вычислительной техники, которые обязаны знать детали и нюансы организации и использования ВМ. К сожалению, по мере развития современных вычислительных систем вопросам изучения аппаратных основ вычислительной техники, знакомству с архитектурой ВМ различных типов на самом низком уровне их организации, на наш взгляд, уделяется недостаточное внимание. Вопросам организации ВМ посвящено значительное количество учебников и специальной литературы [1,2]. Данное учебное пособие и методические указания к циклу лабораторных работ по курсу с обобщенным названием «Организация ЭВМ, микропроцессоры и микропроцессорные системы» дополняет список источников, но в отличие от известных, имеет узкую направленность, поддерживая лабораторный практикум.

Целью данного учебного пособия и методических указаний к лабораторному практикуму по основам соответствующих учебных дисциплин является:

- знакомство с архитектурами микропроцессоров различных типов и микропроцессорных систем на их основе;
- изучение систем команд и микрокоманд микропроцессоров различных типов;
- практическое ознакомление с отдельными БИС микропроцессорных комплектов;
- приобретение навыков программирования и построения вычислительных устройств на основе изучаемых микропроцессоров.

Содержание пособия в определенной степени является расширением и переработкой материалов ранее изданных учебных пособий [2]. В нем рассматриваются некоторые начальные (базовые) положения из курса «Микропроцессорные системы», приводится описание отдельных типов микропроцессоров и принципов построения устройств на их основе. Кроме этого в пособии содержатся методические указания по организации работы на микропроцессорных тренажерах и программе исследований, проводимых в рамках лабораторных работ.

1. Микропроцессоры. Общие сведения

Процессором называется устройство ВМ, непосредственно осуществляющее процесс переработки цифровой информации и управление им в соответствии с заданным алгоритмом, реализованным в виде программы. Процессор занимает центральное место в структуре ВМ. С его помощью осуществляется управление взаимодействием всех устройств, входящих в состав ВМ: он считывает и выполняет команды программы, организует обращение к оперативной памяти, в нужных случаях инициирует работу периферийных устройств, воспринимает и обслуживает запросы прерываний, поступающие из устройств ВМ и извне.

Процессоры современных ВМ в большинстве случаев реализуют на одном кристалле с использованием технологии сверхбольших интегральных схем (СБИС). Соответствующую интегральную схему называют ***микропроцессором*** (МП). Понятие микропроцессор в функциональном отношении совпадает с понятием процессор и отражает лишь особенности, связанные с использованием технологии СБИС при его реализации. Основными достоинствами МП, как одного из наиболее массовых типов средств вычислительной техники, являются программируемость, дешевизна, малые габариты и вес, надежность, простота эксплуатации. Появление МП сыграло революционную роль в автоматике и вычислительной технике, послужив мощным ускорителем развития средств и систем управления и контроля. С созданием микропроцессоров впервые появилась возможность применять вычислительные устройства в таких областях, где раньше это считалось неэффективным либо просто невозможным. Уже первые использования определили два основных направления их применения: использование МП в качестве центральных процессоров вычислительных машин и реализация на базе МП встраиваемых систем управления различными объектами.

Все современные полупроводниковые интегральные микросхемы, в том числе и микропроцессорные, построены на основе транзисторных структур. В зависимости от типа используемых транзисторов различают биполярные и МОП (металл — окисел — полупроводник) микросхемы. На ранних этапах развития микроэлектронных технологий биполярные структуры обеспечивали большую частоту работы, но не допускали высокой плотности размещения элементов на кристалле. В отличие от них канальные (полевые) структуры характеризовались высокой плотностью компоновки элементов, обладали приемлемым быстродействием и более низкой стоимостью производства СБИС. Оба класса микросхем постоянно совершенствовались. В настоящее время обе технологии имеют примерно одинаковые характеристики быстродействия, но КМОП технологии (от англ. CMOS — Complementary Metal Oxide Semiconductor) допускают большую степень интеграции элементов и более технологичны. По этой причине большинство современных микропроцессоров реализуются как КМОП-структуры.

Организация микропроцессоров

Под архитектурой микропроцессора понимается 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. Все названные блоки МП могут размещаться на одном кристалле или на нескольких. Если основные блоки МП размещаются на одном кристалле, то МП называют ***однокристальным,*** в противном случае МП является ***многокристальным***. Типовая структурная схема микропроцессора приведена на рис. 1.1.

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 является 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.

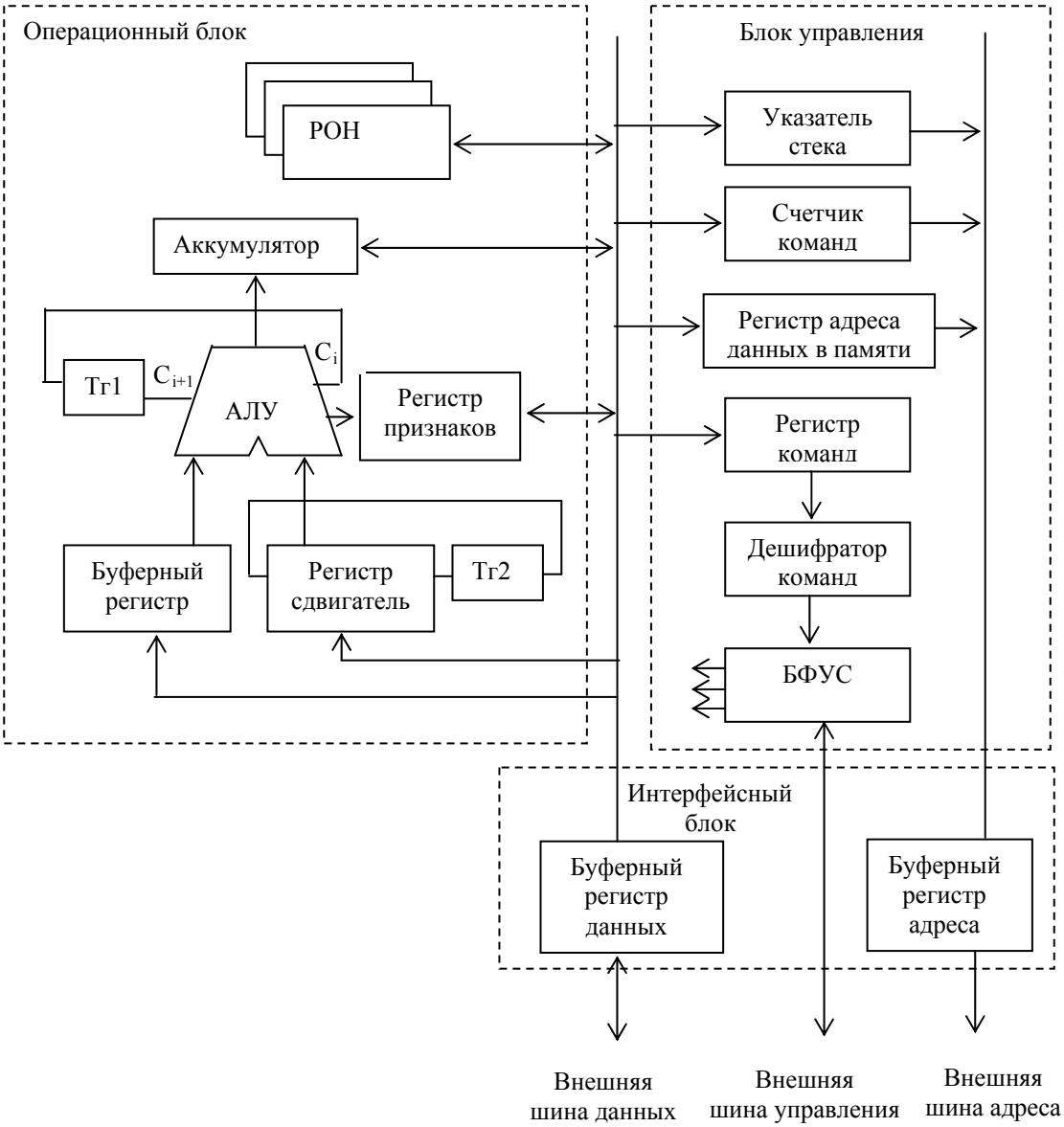


Рис. 1.1. Типовая структура микропроцессора:

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.

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 размещается 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 (рис. 1.1), достаточно широки. С его помощью можно реализовывать различные операции (команды). Например, содержимое любого РОН или ячейки памяти по внутренней шине данных может быть передано в буферный регистр или 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 (рис. 1.1), 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 (заема) 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 разветвлениях 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 стека 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, соответствуют ***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 операций (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 микро-операциях, подлежащих выполнению в течение одного машинного такта, и указание, какая должна быть выбрана из памяти следующая микрокоманда.

Простейший БМУ (рис. 1.2) 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.

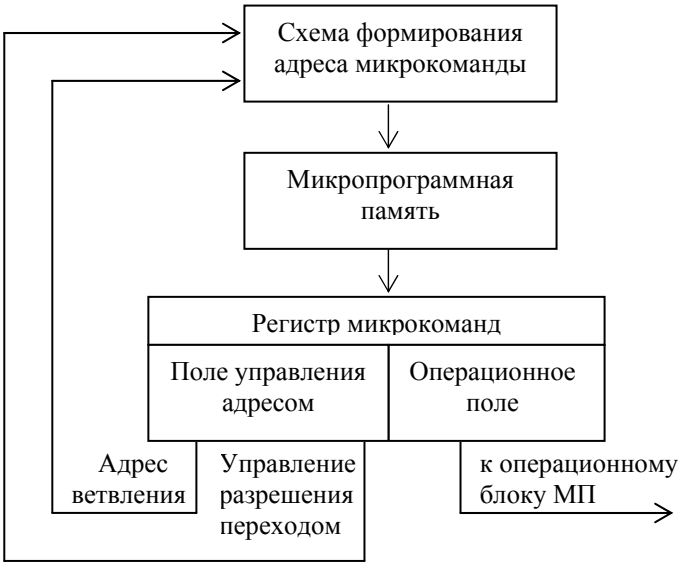


Рис. 1.2. Функциональная схема блока микропрограммного управления

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. Рынок для эмуляторов потенциально огромен.

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 современных микропроцессорах используются оба подхода.

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 функционального 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 позволил 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. Стековая память является ***безадресной***. В качестве указателя стека обычно используют реверсивный счетчик. Принцип работы стека и способ адресации его вершины иллюстрирует рис. 1.3.

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.

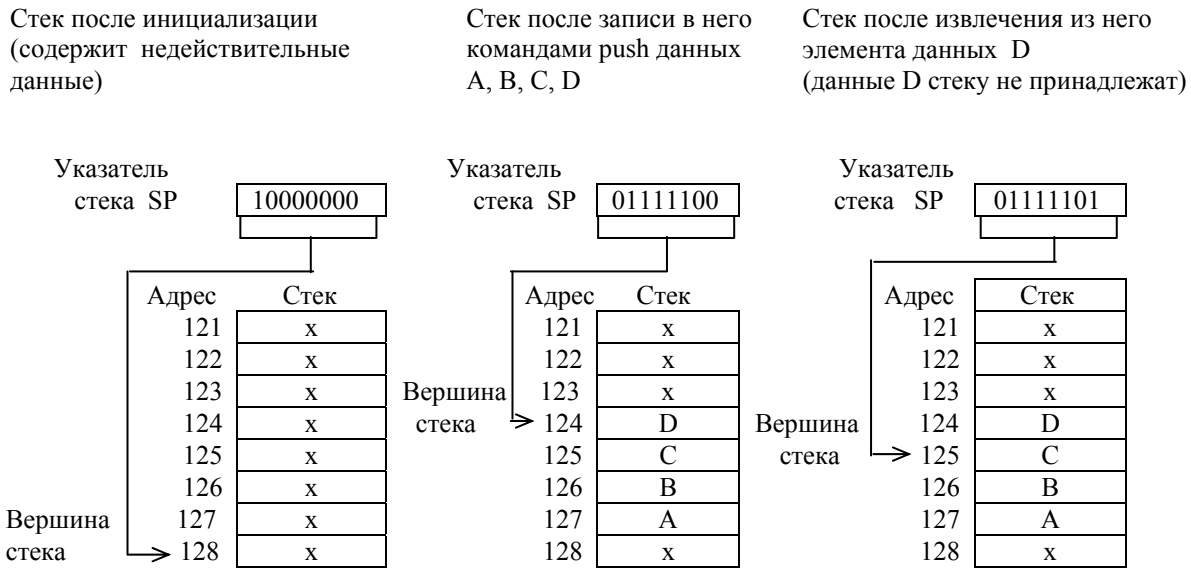


Рис. 1.3. Принцип работы стека и способ адресации его вершины с использованием указателя SP

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 возвратах 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, является 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.

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 группы 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.

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. 1.4). В операционной части команды, состоящей из $n-k$ двоичных разрядов, содержится код операции, обеспечивающий кодирование 2^{n-k} операций и определяющий, какие при этом будут задействованы устройства в МП или вне него. 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 А1,…, А4. Они предназначены для задания кодов адресов операндов (А1, А2), адреса результата (А3) и адреса следующей команды (А4). В качестве адресов А1,…, А3 могут использоваться адреса ячеек оперативной памяти и адреса регистров внутренней памяти МП, а в качестве адреса А4 — только адреса ячеек оперативной памяти. 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 А1 – А4. 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 А4. 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 (разветвлениях, 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 А4, называют ***МП с принудительным способом адресации команд.***

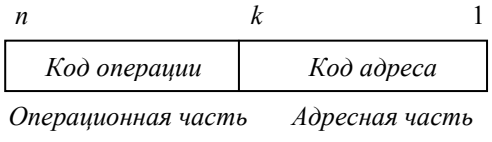


Рис. 1.4. Обобщенная структура команды

Использование адреса результата А3 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, кодирующие 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 другие.

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.

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 $A_к$, 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. 1.5). Прямая адресация используется при работе с простыми переменными и константами, местоположение которых в памяти не меняется в процессе выполнения задачи. При прямой адресации $A_и = A_к$.

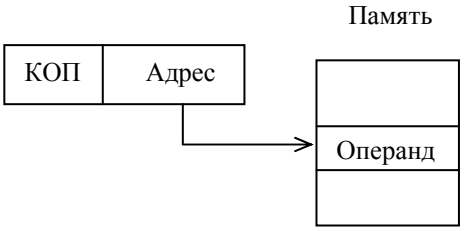


Рис. 1.5. Прямая адресация

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.

Непосредственная адресация позволяет задавать фиксированные значения операнда (Оп) непосредственно в адресной части команды (рис. 1.6). Собственно адресация при этом отсутствует, т. е. Оп = $A_к$, и при обращении к операнду нет необходимости обращаться к памяти.

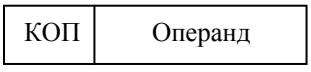


Рис. 1.6. Непосредственная адресация

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.

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 $A_и = (A_к)$. 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. 1.7. 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 косвенная 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.

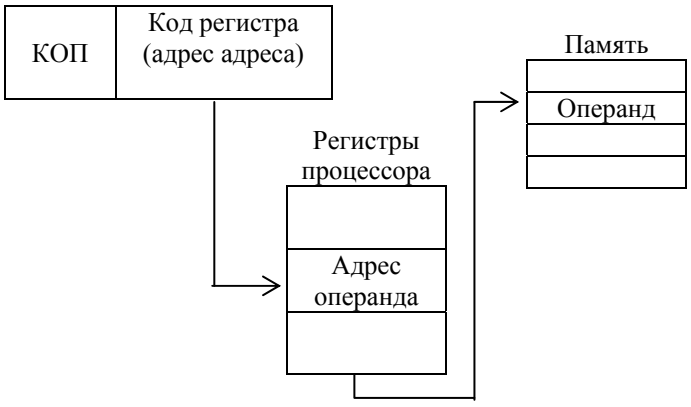


Рис. 1.7. Косвенная адресация

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. 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 (рис. 1.8).

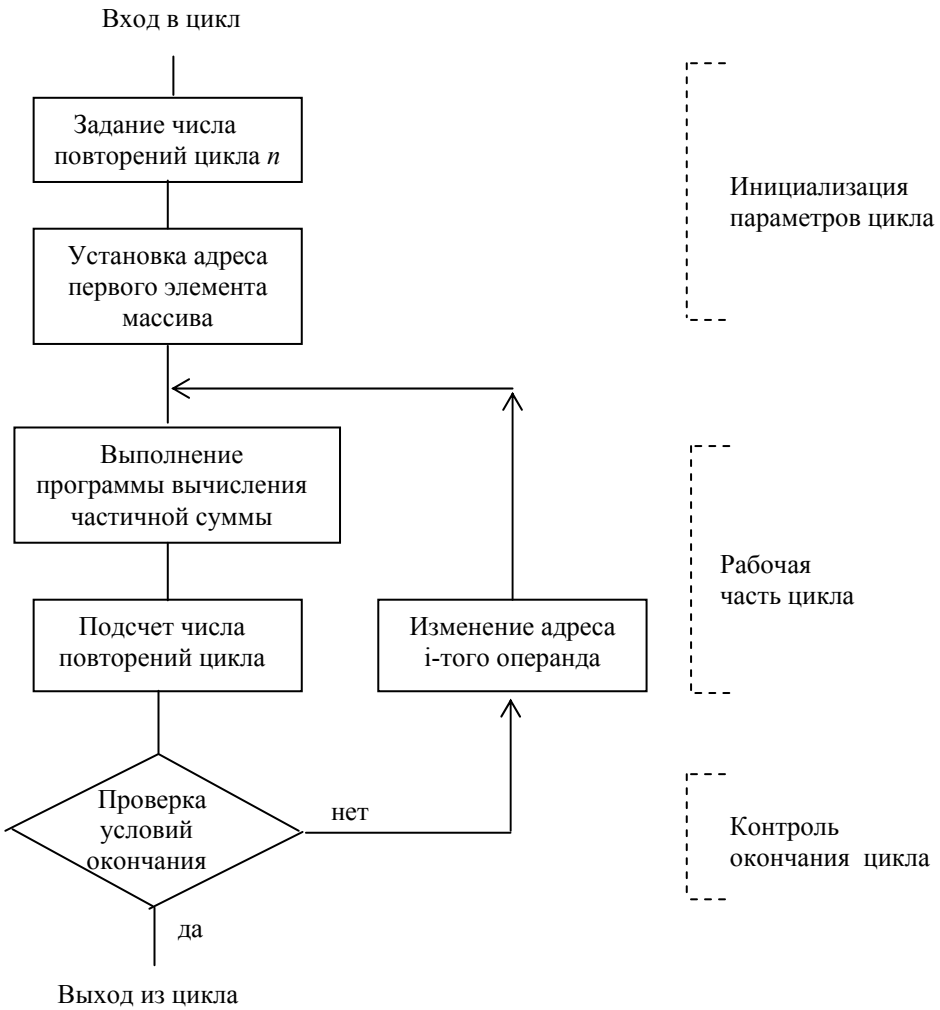


Рис. 1.8. Алгоритм циклической обработки

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. 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.

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. 1.9). Поскольку вычисление исполнительного адреса A_u связано с потерей времени, часто для определения адреса A_u 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.

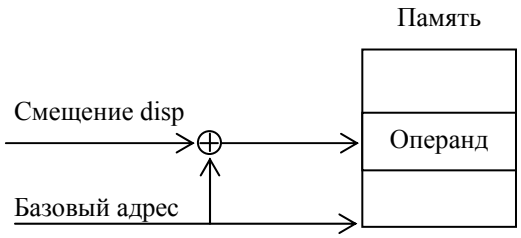


Рис. 1.9. Формирование исполнительного адреса при относительной адресации

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 содержимое 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 (СК), а смещение disp, указываемое в команде, определяет адрес перехода относительно текущего значения счетчика команд

$$A_u = (\text{СК}) + \text{disp}.$$

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

$$A_и = (I) + disp.$$

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

$$A_и = (B) + disp.$$

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 адресации* 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.

2. Однокристальные микропроцессоры с фиксированными разрядностью и списком команд

Особенности микропроцессоров этого класса рассмотрим на примере однокристального 8-разрядного МП КР580ВМ80А с фиксированной системой команд без возможности аппаратного наращивания разрядности. Буква Р в аббревиатуре названия МП указывает, что БИС МП размещается в пластмассовом корпусе. Структурная схема МП КР580ВМ80А (рис. 2.1) 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).

Блок АЛУ обеспечивает выполнение арифметических и поразрядных логических операций, а также операций циклического сдвига. В состав этого блока входят 8-разядное АЛУ, регистр результата — аккумулятор (А), 8-разрядные буферные регистры (БР1 и БР2), регистр признаков (флагов) РФ и схема двоично-десятичной коррекции (СДК). В выполнении арифметических и поразрядных логических операций совместно с АЛУ принимают участие аккумулятор и буферные регистры. Признаки результата указанных операций фиксируются в регистре признаков, содержащем флаги нуля Z, знака S, переноса С, паритета Р и вспомогательного (межтетрадного) переноса АС.

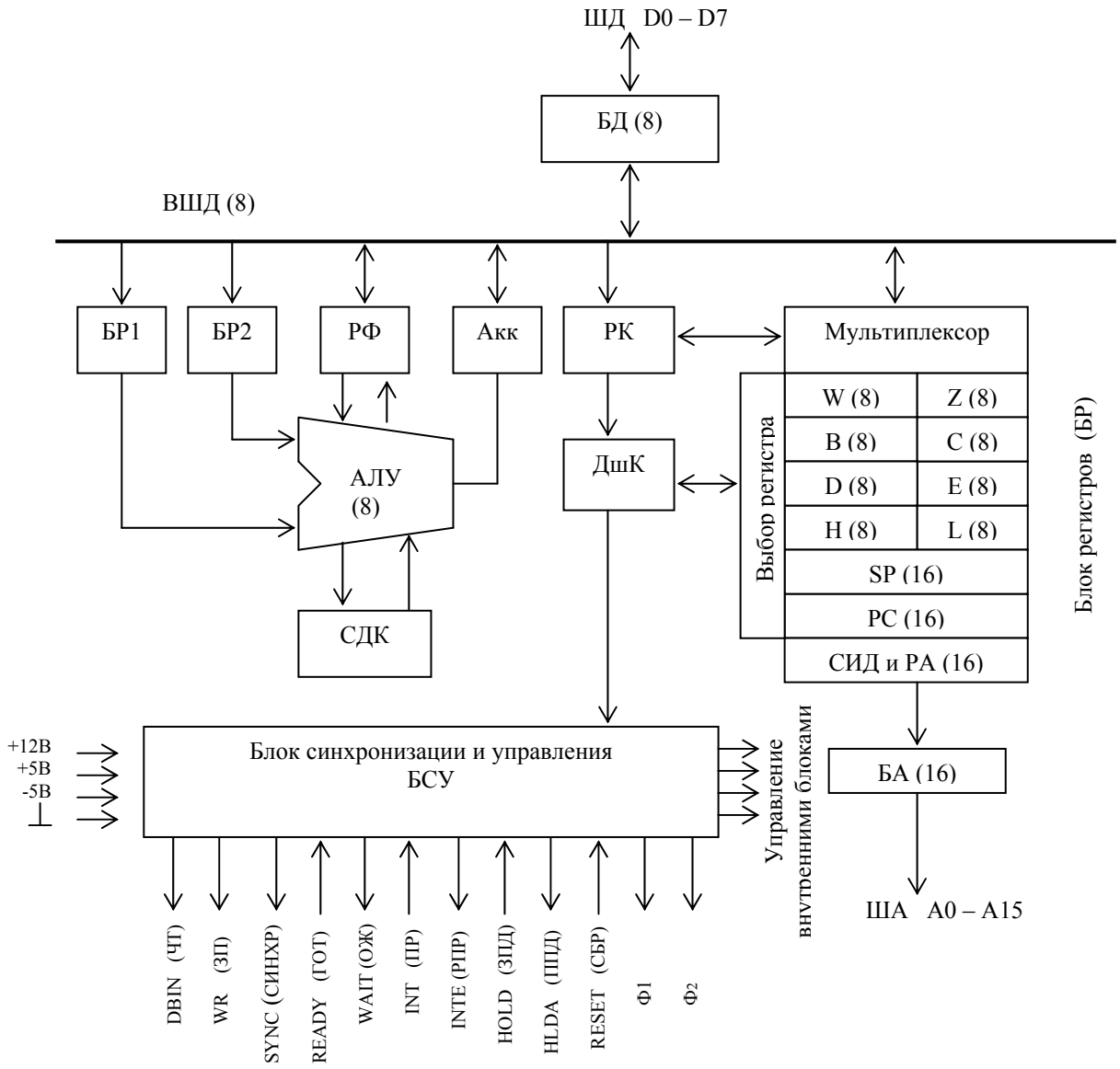


Рис. 2.1. Структурная схема микропроцессора КР580ВМ80А

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 разрядов 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.

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

16-разрядный регистр PA предназначен для запоминания 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 W-Z 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). Обмены 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 реализуются 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. 2.1.

Шина данных объединяет 8 двунаправленных тристабильных линий D7-0. 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 A15-0 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 A15-8 и A7-0. Порты УВВ можно подключать

как к линиям A7-0, так и к линиям A15-8. 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.

Ограниченное число выводов корпуса БИС МП не позволяло непосредственно передавать на выводы микросхемы все необходимые сигналы управления и состояния, определяющие работу МП при выполнении команд.

Для передачи дополнительной информации о предстоящем цикле обмена в первом такте каждого машинного цикла МП выдает на шину данных так называемый байт состояния. Байт состояния, стробируемый сигналом SYNC, записывается во внешний буферный регистр системного контроллера. Отдельные биты байта состояния являются управляющими сигналами. Используя эти сигналы и выходные сигналы микропроцессора WR и DBIN, системный контроллер формирует сигналы «Чтение» и «Запись» отдельно для памяти и отдельно для периферийных устройств ввода/вывода. При реализации всего списка команд используется 10 видов машинных циклов: Типы машинных циклов и соответствующие им сигналы байта состояния приведены в прил. 2.

Функционирование микропроцессора

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. В такт Т1 осуществляется адресация внешнего устройства, к которому выполняется обращение. В такт Т2 реализуется проверка управляющих сигналов READY, INT, HOLD, влияющих на функционирование МП. В такт Т3 выполняется собственно внешний обмен. Такты Т4 и Т5 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 mov R_D,R_S из регистра источника R_D в регистр приемник R_S 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.

Знание особенностей структуры МП, 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 МП 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.

Программная (регистровая) модель микропроцессорной системы на базе МП КР580ВМ80А (рис. 2.2) включает только программно доступные регистры.

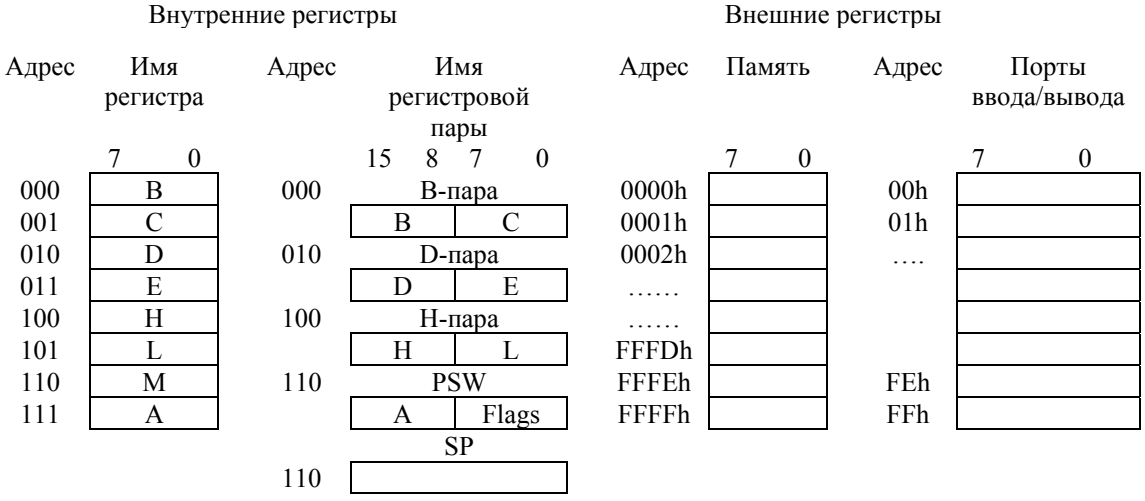


Рис. 2.2. Программная модель микропроцессорной системы с МП КР580ВМ80А

В состав программной модели системы на базе МП КР580ВМ80А входят: 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 64 К. При обращении к памяти используется 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.

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). Адреса регистров и регистровых пар приведены в табл. П1 прил. 1.

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. 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* 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 (рис. 2.2). 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*).

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. В частности, двухбайтовые команды непосредственной загрузки регистров с мнемоникой *MVI* R_D*,B2* 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 (рис. 2.2). Другие команд группы пересылок менее однородны по составу, кодирование этих команд труднее поддается формализации.

Двухбайтные команды ввода *IN port* и вывода *OUT port* 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.

Обмен данными между памятью и аккумулятором также может осуществляться с помощью команд загрузки с мнемоникой *LDA* и запоминания с мнемоникой *STA*. 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*. В качестве указателей памяти в этих командах используют пары регистров B и D.

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. В качестве второго операнда может использоваться содержимое любого регистра блока РОН или ячейки памяти М, 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. Обобщенный код команд указанной группы имеет вид **2XS**, где 2 — код команд арифметических и поразрядных логических операций; X — код типа арифметических и поразрядных логических операций (табл. 2.1); S (SSS) — код регистра источника. Коды (адреса) регистров S указаны в поле адреса регистра регистровой модели МП (см. рис. 2.2).

Таблица 2.1

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

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 заемом *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 флаге переноса C. Перестановки элементов массива (собственно упорядочение) производят в соответствии со значением флага C.

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 флага C, который принудительно сбрасывается в 0.

Команду *ANA R*, осуществляющую поразрядную конъюнкцию операндов (табл. 2.1), применяют для проверки значения определенного бита в байте, содержащемся в аккумуляторе, с помощью другого байта-маски. В частности, если требуется проверить состояние бита A_3, то необходимо использовать маску 00001000. О состоянии бита 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 (табл. 2.1). 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 выполнении 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 имеют обобщенный код команд **3X6 B2**, где X — код типа арифметических и поразрядных логических операций (табл. 2.1). Операндами этих команд являются содержимое аккумулятора и второй байт команды B2.

Однобайтные команды инкремента *INR R* ($r \leftarrow r + 1$) и декремента *DCR R* ($r \leftarrow r - 1$), имеют обобщенный код **0D4** (инкремента) и **0D5** (декремента), где D — код регистра приемника. 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.

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 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.

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. 2.3. 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 — тип команды сдвига:

0 — *RLC*, 1 — *RRC*, 2 — *RAL*, 3 — *RAR*.

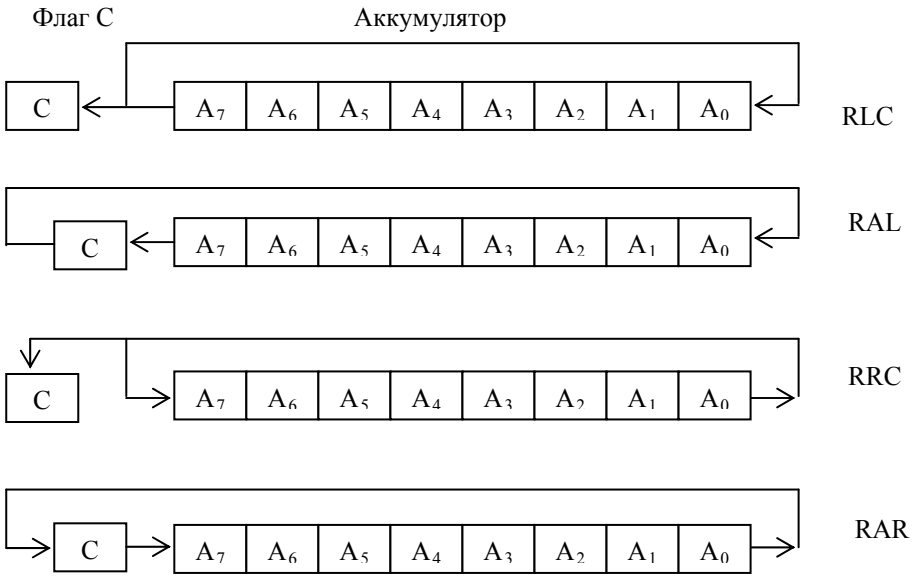


Рис. 2.3. Команды сдвига

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* 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. Трехбайтная команда вызова подпрограмм *CALL addr* 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*, перезагружающей содержимое PC адресом возврата. Команда *CALL addr* выполняется за пять машинных циклов, а команда *RET* — за три машинных цикла.

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

M8 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. 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 $000\ 000_8$, $000\ 010_8$, $000\ 020_8$, …, $000\ 070_8$. Подпрограмма должна завершаться командой возврата *RET*. Команда вызова *RST n* выполняется за три машинных цикла.

Однобайтная команда пересылки 16-битных операндов *PCHL* по выполняемым функциям является особой командой безусловного перехода. При ее выполнении содержимое регистровой 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.

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 значению каждого из флагов, кроме флага AC.

Обобщенный код команд условной передачи управления имеет вид **3YZ**, где 3 — код класса команд передачи управления, Y — код проверяемого условия (табл. 2.2), Z — код одной из трех групп команд условной передачи управления (табл. 2.3).

Всего имеется 24 команды условной передачи управления, проверяющие единичное и нулевое значение 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.

Таблица 2.2

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
Нечетный	PO	0	100
Четный	PE	1	101
Положительный	P	0	110
Отрицательный	M	1	111

Таблица 2.3

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 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 (рис. 2.2).

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 РФ (рис. 2.4).

Содержимое аккумулятора A

A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0

Содержимое регистра флагов РФ

S	Z	0	AC	0	P	1	C

Рис. 2.4. Слово состояние программы PSW

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 процессора PSW.

Восьмеричный код команд *PUSH rp* и *POP rp* — **3rp5** и **3rp1** соответственно.

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 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.

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. Команды двойного сложения *DAD rp* выполняют 16-битное суммирование содержимого HL с содержимым адресуемой регистровой пары BC, DE, HL или SP. 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 реализует обмен содержимого регистровых пар DE и HL:

```
XCHG
DAD H
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 устанавливают/ сбрасывают внутренний триггер разрешения прерываний INTE. В состоянии INTE = 1 микропроцессор реагирует на внешние запросы прерываний, при INTE = 0 — прерывания запрещены.

По команде останова *HLT* 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 ОЖ. 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 выполнению 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».

Холостая команда NOP не производит никаких действий, кроме инкремента счетчика команд.

Полный список команд МП КР580ВМ80А приведен в прил. 1.

3. Секционные микропроцессоры

В многокристальных МП операционный блок конструктивно отделен от управляющего. В секционных МП этот блок разбивается на идентичные модули, реализуемые в виде отдельных БИС (микропроцессорные секции). Поскольку число секций, определяющих разрядность обрабатываемых слов, не ограничена, объединение секций под общим микропрограммным управлением позволяет наращивать разрядность МП. Это свойство секционных микропроцессоров является их отличительной характеристикой.

Типовым представителем секционных МП является микропроцессор, реализованный на основе БИС с разрядно-модульной организацией, входящих в состав микропроцессорного комплекта К1804 (МПК К1804). Базовый комплект содержит 9 БИС, выполненных по ТТЛДШ технологии с задержкой распространения сигнала от 10 нс до 100 нс. 5 БИС (К1804ВС1, К1804ВС2 — центральные процессорные элементы — ЦПЭ, К1804ВР1, К1804ВР2, реализующие ускоренные переносы и управляющие состоянием и сдвигами, а также четырехразрядный регистр К1804ИР1) ориентированы для построения операционных блоков микропроцессоров. Остальные БИС комплекта (К1804ВУ1, К1804ВУ2, К1804ВУ3 и К1804ВУ4) предназначены для построения блоков микропрограммного управления.

Четырехразрядная микропроцессорная секция (ЦПЭ) К1804ВС1

ЦПЭ предназначен для арифметико-логической обработки данных и временного хранения операндов и результатов вычислений во внутренней регистровой памяти. Структурная схема секции ЦПЭ содержит четыре основных блока: блок АЛУ, блок внутренней памяти (регистровое ЗУ), блок регистра Q и блок управления (рис. 3.1).

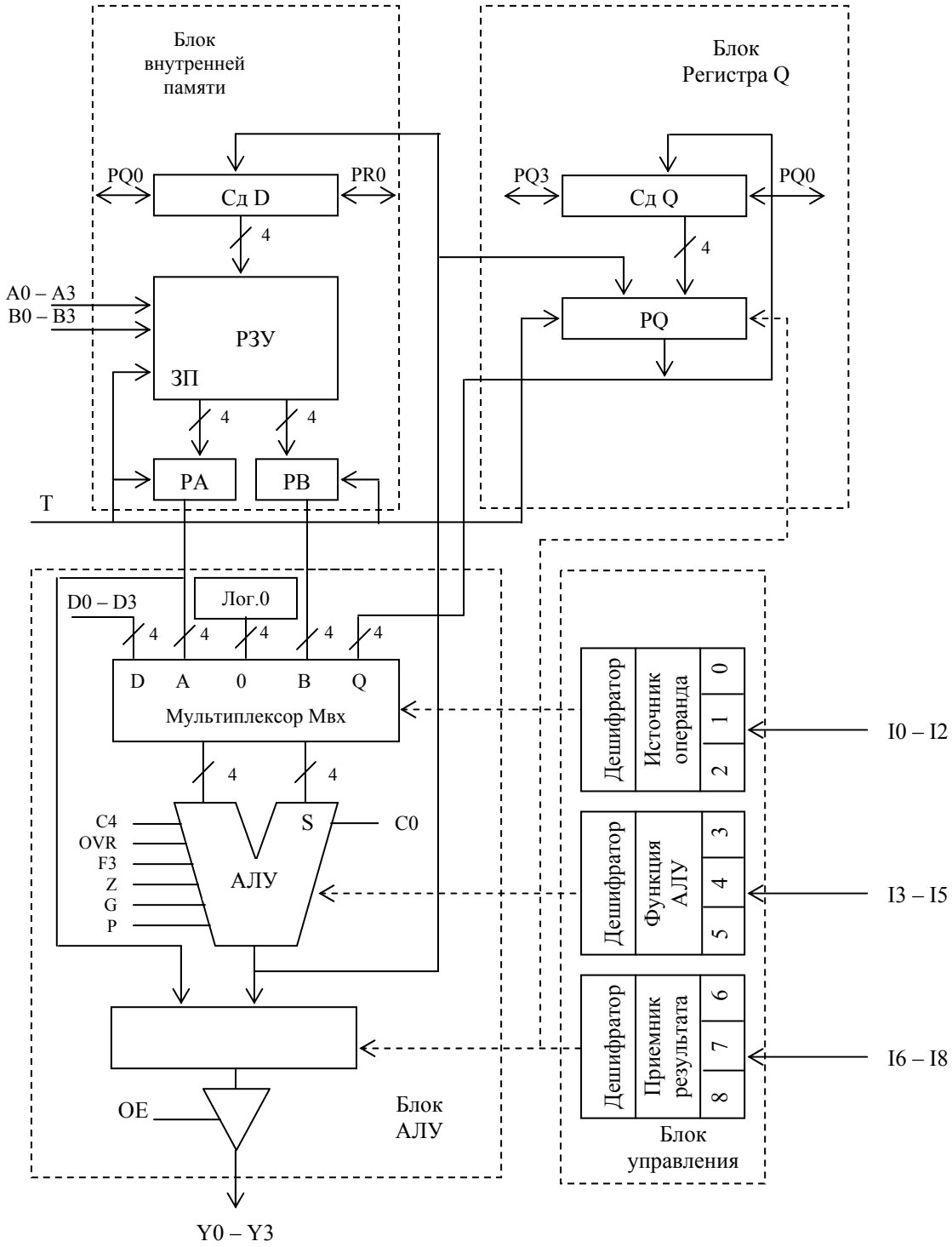


Рис.3.1.Структурная схема четырехразрядной микропроцессорной секции К1804ВС1

ЦПЭ управляется операционным полем микрокоманды (рис. 3.2). Оно содержит 9-разрядный микрокод управления блоком АЛУ (разряды I8–I0), *n*-разрядное поле данных D текущей микрокоманды, два 4-разрядных адресных поля A и B регистров регистрового ЗУ, используемых в текущей операции, а также необязательные дополнительные поля, наличие которых определяет непосредственно пользователь. В число таких полей, например, могут входить входной перенос C0, поле OE, управляющее разрешением вывода информации с выхода ЦПЭ на выходную шину данных Y и другие. 9-разрядный микрокод

управления блоком АЛУ определяет, какую операцию необходимо выполнить в текущей микрокоманде (микрокод I5 I4 I3), где размещаются операнды выполняемой операции (микрокод I2 I1 I0) и куда помещается результат (микрокод I8 I7 I6). Кодировка этих полей микрокоманды рассматривается ниже при описании блока АЛУ.

	Поле входных данных D	Микрокод управления блоком АЛУ	Адресное поле B регистрового ЗУ	Адресное поле A регистрового ЗУ
OE . . . C0	D3…D0	I8 . . . I0	B3 . . . B0	A3 . . . A0

Рис. 3.2. Формат микрокоманды секции ЦПЭ

Арифметико-логический блок содержит четырехразрядное АЛУ, селектор источников входных данных и селектор выходных данных. АЛУ выполняет простейшие арифметические и поразрядные логические операции и формирует четыре признака результата: перенос из старшего разряда C4, переполнение OVR, знак результата F3 и признак нулевого результата Z. С помощью признака переполнения OVR контролируется правильность выполнения операций над знаковыми числами: если числа представлены со знаками, то при OVR=1 искажается знаковый разряд, т. е. результат оказывается ошибочным. Признаки результата в большинстве случаев запоминаются в специальном внешнем регистре состояния и используются при реализации условных переходов в микропрограммах. При выполнении операций АЛУ также формируются сигналы генерации G и распространения P переноса из АЛУ, необходимые для организации ускоренного переноса в многоразрядном процессоре, построенном из нескольких секций ЦПЭ.

Арифметические операции в АЛУ выполняются с учетом значения входного переноса C_0 по правилам дополнительного кода. Логические операции реализуются поразрядно. Выполняемая АЛУ операция определяется набором управляющих сигналов, формируемых дешифратором блока управления ЦПЭ из сигналов I5 I4 I3 микрокоманды (табл. 3.1). За 1 тактовый синхроимпульс над входными данными в АЛУ может быть выполнена некоторая операция и полученный результат записан в регистровое ЗУ без сдвига или со сдвигом вправо или влево на один разряд.

Таблица 3.1

Микрокод операции АЛУ			Реализуемая операция
I5	I4	I3	
0	0	0	R + S + C0
0	0	1	S - R – 1 + C0
0	1	0	R - S – 1 + C0
0	1	1	R $\vee$ S
1	0	0	R ^ S
1	0	1	$\overline{R}$ ^ S
1	1	0	R $\oplus$ S
1	1	1	$\overline{R \oplus S}$

На входы R и S АЛУ операнды могут поступать от пяти возможных источников входных данных. В качестве последних могут выступать константа «0», содержимое буферных регистров A и B блока регистрового ЗУ, содержимое регистра Q и входные данные D0 — D3. Выбор источников операндов, подключаемых к входам R и S АЛУ, осуществляет мультиплексор, управляемый полем I2 I1 I0 кода микрокоманды (табл. 3.2).

Таблица 3.2

Микрокод выбора источников операндов			Источник операнда	
I2	I1	I0	R	S
0	0	0	A	Q
0	0	1	A	B
0	1	0	0	Q
0	1	1	0	B
1	0	0	0	A
1	0	1	D	A
1	1	0	D	Q
1	1	1	D	0

Здесь: A, B — содержимое регистров, адресуемых в адресных полях A и B микрокоманды; Q — содержимое дополнительного регистра Q; D — значение данных на входе ЦПЭ; 0 — константа «логический нуль».

В тех случаях, когда на вход D информация поступает от нескольких источников, одним из которых является константа, возникает необходимость коммутации этих источников с помощью внешнего мультиплексора. Для хранения константы в микрокоманде обычно выделяется специальное поле, число разрядов которого равно разрядности обрабатываемых данных (отдельные четверки разрядов 4*n*-разрядного поля константы D подаются в соответствующие секции ЦПЭ). Управление коммутированием источников входных данных (рис. 3.3) осуществляется разрядами дополнительного поля микрокоманды, которые подключаются к адресным входам мультиплексора.

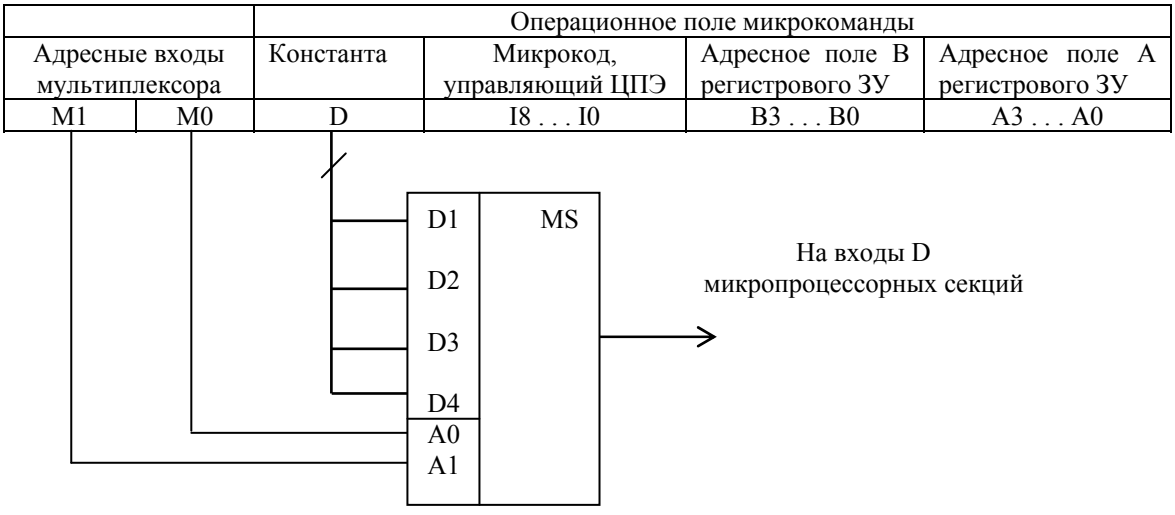


Рис. 3.3. Схема коммутации внешних источников данных на входы D ЦПЭ

С выхода АЛУ результат выполненной операции подается на селектор выходных данных. На второй вход селектора информация поступает с выхода буферного регистра A, минуя АЛУ. Информация с выхода селектора через управляемые усилители передается на выходную трехстабильную шину Y ЦПЭ. Управляет селектором выходных данных поле I8 I7 I6 микрокоманды.

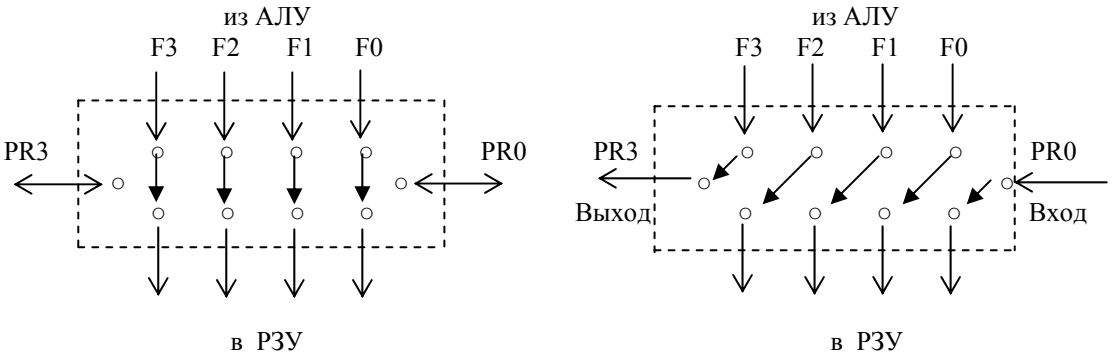
Внутри ЦПЭ приемником результата операции, выполненной в АЛУ, может быть регистр блока РОН, адресуемый полем B микрокоманды, или регистр Q. Информация с выхода АЛУ может быть либо непосредственно записана в

регистровое ЗУ, либо перед записью сдвинута влево или вправо на один разряд, что соответствует выполнению операций вида 2F или F/2. Приемник и механизм загрузки результата выполненной в АЛУ операции определяется разрядами I8 I7 I6 микрокоманды (табл. 3.3).

Таблица 3.3

Двоичный код, задающий приемник результата			Регистровое ЗУ		Регистр Q		Выход Y ЦПЭ
I8	I7	I6	Сдвиг	Загрузка	Сдвиг	Загрузка	
0	0	0	–	–	–	F → Q	F
0	0	1	–	–	–	–	F
0	1	0	–	F → B	–	–	A
0	1	1	–	F → B	–	–	F
1	0	0	Вправо	F/2 → B	Вправо	Q/2 → Q	F
1	0	1	Вправо	F/2 → B		–	F
1	1	0	Влево	2F → B	Влево	2Q → Q	F
1	1	1	Влево	2F → B		–	F

П р и м е ч а н и я:
F — выход АЛУ; – — отсутствие операции сдвига или загрузки;
A — выход регистра, адресуемого полем A; B — регистр-приемник результата, адресуемый полем B;
Q - регистр-приемник результата Q.



а) Запись в РЗУ без сдвига

б) Запись в РЗУ со сдвигом влево

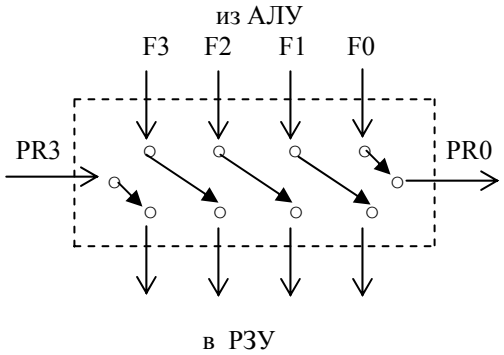


Рис. 3.4. Способы записи информации с выхода АЛУ в регистровое ЗУ: а) без сдвига (непосредственная запись); б) со сдвигом влево на один разряд; в) со сдвигом вправо на один разряд

Операции сдвига вправо или влево данных с выхода АЛУ реализуются с помощью сдвигателя, выполненного в виде мультиплексора на входе регистрового ЗУ. Мультиплексор осуществляет либо прямую передачу данных из АЛУ в регистровое ЗУ, либо передачу данных со сдвигом влево или вправо.

Особенностью МП К1804ВС1 является то, что в нем выдвигаемые при сдвигах биты не фиксируются во флаге переноса С. Это связано с тем, что сдвиговые операции в данном МП реализуются вне АЛУ. Способы записи информации с выхода АЛУ через мультиплексор в регистровое ЗУ показаны на рис 3.4.

Блок внутренней памяти состоит из регистрового ЗУ емкостью шестнадцать 4-разрядных РОН со схемами записи и считывания, двух буферных регистров А и В, хранящих содержимое регистров, адресуемых одноименными входами А и В, и регистра сдвигателя на входе регистрового ЗУ. Адресная информация (4-разрядные коды А и В), определяет адресуемые в текущей микрокоманде регистры блока РОН. Эта информация содержится в одноименных полях операционного поля микрокоманды, управляющей работой ЦПЭ. Запись в регистровое ЗУ возможна только по адресу В.

Чтение и запись в регистровое ЗУ разнесены во времени. Чтение данных из ЗУ происходит при единичном логическом уровне тактового синхроимпульса. При чтении вход регистрового ЗУ отключается от сдвигателя, и регистровое ЗУ не реагирует на входную информацию. Запись в ЗУ осуществляется при нулевом логическом уровне тактового синхроимпульса, при этом выход регистрового ЗУ отключается от входов буферных регистров А и В, которые продолжают хранить ранее записанную информацию. Двунаправленные входы/выходы сдвигателя PR0 и PR3 в зависимости от направления сдвига служат входом или выходом, через которые производится запись в освобождающийся при сдвиге разряд и выдача содержимого выдвигаемого разряда. При объединении нескольких секций ЦПЭ вывод PR3 младшей (i – 1)-й секции должен быть соединен с выводом PR0 старшей i-й секции.

Блок регистра Q состоит из дополнительного 4-разрядного регистра Q и отдельного сдвигателя, позволяющего сдвигать содержимое регистра Q влево или вправо на один разряд. Управление блоком регистра Q осуществляется разрядами поля микрокоманды I8 I7 I6 (табл. 3.3). На вход регистра Q могут поступать либо результат операции АЛУ, либо данные с выхода собственного сдвигателя. Операции сдвига содержимого регистра Q производятся параллельно с операциями сдвига в регистровом ЗУ.

Блок управления (рис. 3.1) формирует сигналы управления работой секции ЦПЭ. В его состав входят три дешифратора, с помощью которых микрокод I8 – I0 преобразуется в набор сигналов, обеспечивающих выполнение требуемой операции (разряды I5 I4 I3), выбор источников операндов выполняемой операции (разряды I2 I1 I0) и выбор приемника результата (разряды I8 I7 I6).

Объединение секций ЦПЭ в операционном устройстве 4*n*-разрядного процессора. Для построения 4n-разрядного процессора необходимо соединить цепи межразрядных переносов n секций ЦПЭ и объединить их общей шиной микропрограммного управления. При этом требуется решить, по крайней мере, три задачи:

– обеспечить малое время задержки распространения волны переносов, поступающих на вход С0 микропроцессорных секций, при выполнении арифметических операций;

– реализовать цепи передачи межсекционных переносов сдвиговых операций;

– сформировать признаки результата операций 4*n*-разрядного процессора.

При последовательном соединении секций, когда выход C4 младшей секции подключается к входу C0 следующей секции, задержки распространения волны последовательных переносов могут оказаться значительными и существенно повлиять на быстродействие операционного устройства в целом. Для уменьшения задержки в формировании и передачи переносов используют специальные схемы ускоренного переноса, например, БИС К1804ВР1. Одна схема К1804ВР1 позволяет организовать параллельные цепи переноса в операционном блоке, содержащем до четырех 4-разрядных секций ЦПЭ. При разрядности процессора больше 16 необходимо использовать каскадное включение БИС К1804ВР1. В качестве сигналов, необходимых для формирования ускоренных переносов, используются сигналы с выходов P и G микропроцессорных секций К1804ВС1.

Другая задача, решаемая при объединении микропроцессорных секций, состоит в построении цепей передачи переносов при выполнении сдвиговых операций. При сдвиге вправо биты младших разрядов, выдвигаемые из старших секций на выходы PR0 и PQ0, должны передаваться на входы PR3 и PQ3 следующих младших секций для ввода их в освобождающиеся при сдвиге старшие разряды регистров. При сдвиге влево на выходы PR3 и PQ3 выдвигаются биты старших разрядов секции, которые должны вдвигаться через входы PR0 и PQ0 в освобождающиеся при сдвиге младшие разряды регистров старших секций. Следовательно, при объединении микропроцессорных секций необходимо обеспечить соединение выводов PR3 и PQ3 младшей секции с выводами PR0 и PQ0 старшей секции, при этом свободными оказываются выводы PR3 и PQ3 старшей секции и выводы PR0 и PQ0 младшей секции. Для организации различных типов сдвигов (логических, арифметических, вправо, влево, циклических, одинарной или двойной длины) каждый тип сдвига требует определенного соединения свободных выводов младшей и старшей секций 4*n*-разрядного операционного блока (при организации циклических сдвигов) или подачи на свободные выводы определенных сигналов (при организации других типов сдвигов). Такое соединение свободных выводов секций ЦПЭ может быть выполнено с помощью мультиплексоров, управляемых сигналами, задаваемыми в специальном (дополнительном) поле микрокоманды.

Третьей задачей, решаемой при построении многоразрядного секционного операционного блока, является формирование слова состояния программы — признаков результата операции, выполненной 4*n*-разрядным процессором. Для формирования признака нуля Z секционного микропроцессора необходимо объединить выводы Z секций и подключить их через резистор к источнику питания. В качестве признаков C4, OVR и F3 используются одноименные признаки, формируемые на выходах старшей микропроцессорной секции. Выходы признаков остальных секций остаются неиспользованными.

Блок микропрограммного управления

Простейший БМУ, состоящий из схемы формирования адресов микрокоманд и микропрограммной памяти с выходным регистром микрокоманд, рассмотрен в разд. 1 (рис. 1.2).

На базе 4-разрядных секций управления адресом К1804ВУ1 и К1804ВУ2 (СУАМ) можно реализовать БМУ с произвольной разрядностью адреса микрокоманд. Наращивание разрядности адреса МК осуществляется за счет объединения нескольких секций управления адресом по цепям переноса.

БИС МПК К1804 позволяют строить БМУ с широкими логическими возможностями. 12-разрядная схема управления последовательностью микрокоманд К1804ВУ4 предназначена для генерации последовательности адресов микрокоманд, хранящихся в микропрограммной памяти объемом 4096 слов. БИС К1804ВУ4 не позволяет наращивать разрядность за счет объединения нескольких БИС и этим отличается от 4-разрядных секций управления адресом К1804ВУ1 и К1804ВУ2.

Секции управления адресом микрокоманды К1804ВУ1

Основным назначением секции управления адресом микрокоманды (СУАМ) (рис. 3.5) является формирование адреса следующей микрокоманды.

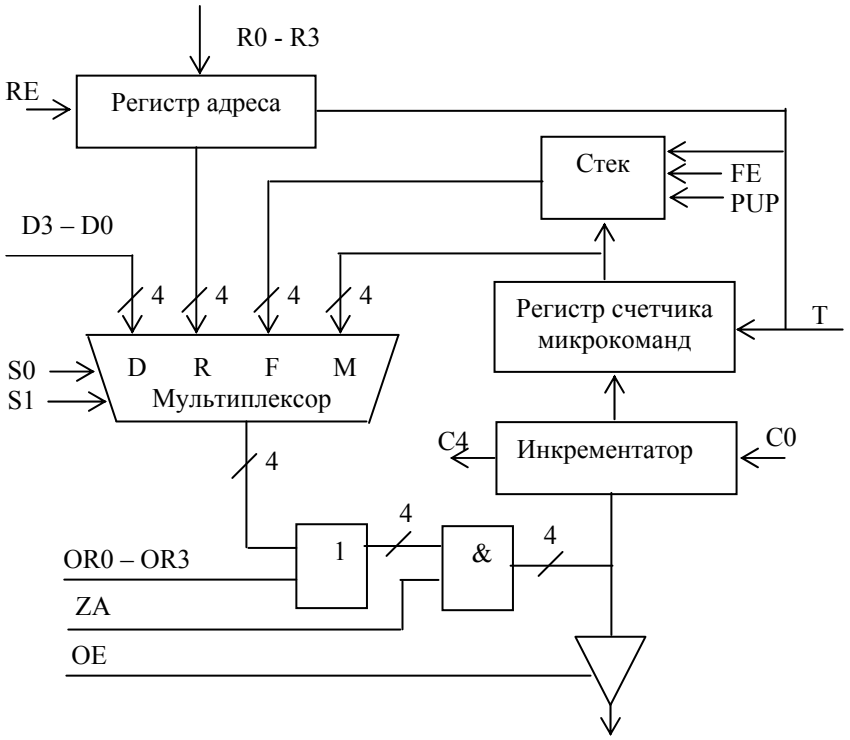


Рис. 3.5. Структурная схема СУАМ К1804ВУ1

Источником адреса микрокоманд могут быть как внутренние блоки СУАМ, так и внешние формирователи адреса микрокоманд. К внутренним блокам, в первую очередь, относятся счетчик микрокоманд, обеспечивающий последовательное считывание команд, и стек, в котором запоминаются адреса возврата при работе с подпрограммами. Внешними источниками адреса

являются регистр микрокоманд (его адресная часть) и преобразователи начального адреса микропрограмм команд и векторов прерываний.

Источник адреса в СУАМ выбирается с помощью четырехвходового мультиплексора. Источником адреса могут быть стек, счетчик микрокоманд, адресная информация, поступающая на входы СУАМ D0-D3, либо регистр адреса. Для управления мультиплексором используются сигналы мультиплексора S0 и S1, указываемые в адресном поле микрокоманды. На выходе мультиплексора включены схемы ИЛИ, позволяющие модифицировать выбранный адрес с помощью маски, подаваемой входы OR3-OR0. Формируемый адрес с помощью схем И может быть замаскирован сигналом ZA. При ZA=0 на выходе СУАМ устанавливается нулевой код независимо от содержимого выбранного источника адреса. С выхода схем И адрес через усилители, управляемые сигналом OE, передается на выходную тристабильную шину СУАМ Y3 – Y0. Кратко охарактеризуем внутренние источники адреса СУАМ.

Счетчик микрокоманд состоит из регистра счетчика микрокоманд (РгСчМК) и схемы приращения — инкрементатора. Любой текущий адрес с выхода мультиплексора через инкрементатор передается в РгСчМК для запоминания. Адрес, формируемый на выходе РгСчМК, непосредственно зависит от значения входного переноса C0. При C0 = 0 адрес на выходе РгСчМК не изменяется, что позволяет обеспечить выполнение одной и той же микрокоманды любое заданное число раз. При C0 = 1 происходит увеличение адреса на выходе РгСчМК, и обеспечивается последовательное считывание микрокоманд из микропрограммной памяти.

Аппаратный стек СУАМ. В отличие от стека, моделируемого в памяти, в БМУ МП К1804 используется аппаратный стек. Основным преимуществом аппаратного стека является существенно меньшее время обращения к стеку. Аппаратный стек реализуется в виде набора регистров, размещенных на кристалле, вместе с другими элементами СУАМ К1804ВУ1. Важным параметром аппаратного стека является его глубина — число регистров стека. При попытке записать в стек большее количество слов, чем число регистров стека, первое записанное слово будет утеряно. В БИС К1804ВУ1 используется 4-уровневый стек, состоящий из указателя стека SP, дешифратора, накопителя и схемы записи/считывания. Указатель стека представляет собой реверсивный счетчик. Выходной код указателя стека, преобразуемый дешифратором в адрес регистра накопителя, определяет регистр накопителя, к которому производится обращение. Указатель SP всегда определяет последнее записанное в него слово. Схема записи/считывания обеспечивает необходимую коммутацию для передачи информации из счетчика микрокоманд в накопитель стека (при записи в стек) или из накопителя стека на вход мультиплексора (при считывании из стека). Стек управляется двумя внешними входами — разрешения работы стека FE и управления загрузкой/выгрузкой стека PUP. При выполнении операций со стеком на указанные входы подаются одноименные сигналы с выхода схемы формирования сигналов управления СУАМ, реализованной в виде ПЗУ емкостью 32 восьмиразрядных слова. При реализации

подпрограмм стек обеспечивает запоминание адреса возврата при переходе к первой микрокоманде подпрограммы (FE = 0 PUP = 1) и автоматический возврат в прерванную точку программы по окончании подпрограммы (FE = 0 PUP = 0). Комбинации значений входных сигналов FE и PUP и определяемые ими режимы работы стека представлены в табл. 3.4.

Таблица 3.4

FE	PUP	Режим работы стека
0	0	Выбор адреса возврата из стека с последующим декрементом SP
0	1	Инкремент SP с последующей загрузкой в стек текущего содержимого счетчика микрокоманд
1	x	Выполнение операций, не связанных со стеком

Прямые входы адреса D3 – D0 используются для непосредственной передачи адресной информации от внешних источников на выход СУАМ.

Внешними источниками адреса, как отмечено выше, могут являться регистр микрокоманд (его адресная часть) и преобразователи начального адреса микропрограмм команд и векторов прерываний. В частности, с помощью этих входов адрес перехода из регистра микрокоманд поступает на вход мультиплексора адреса. Поясним назначение других источников внешнего адреса.

Преобразователи начального адреса микропрограмм используются в системах, в которых применяется двухуровневое программирование секционных МП с микропрограммным управлением — на языке команд и на языке микрокоманд. В таких системах пользовательские программы, как правило, пишутся не с использованием микрокоманд, хотя это и не исключается, а на выбранном или заданном языке программирования, при этом команды программы размещаются в блоке внешней программной памяти. Команды адресуются специальным счетчиком команд, который обычно реализуется на одном из внутренних регистров ЦПЭ. Каждая команда представляется последовательностью микрокоманд — микропрограммой команды. Микропрограммы всех команд хранятся в микропрограммной памяти, адресуемой БМУ. Последней микрокомандой любой команды является микрокоманда обращения к счетчику команд, обеспечивающая переход к следующей команде программы. Идентификация команд осуществляется по коду команды, который всегда определяет адрес первой микрокоманды команды в микропрограммной памяти. При выборе команды из внешней программной памяти код команды поступает в специальный преобразователь начального адреса микропрограмм, который преобразует его в адрес первой микрокоманды микропрограммы считанной команды. По завершению выполнения последней микрокоманды очередной команды осуществляется инкремент содержимого счетчика команд и переход к следующей команде программы. Аналогично функционирует преобразователь начального адреса микропрограмм векторов прерывания.

Регистр адреса применяется в качестве буфера для хранения адреса, поступающего от внешнего источника на входы R3 – R0. Разрешение записи осуществляется сигналом RE = 0. С помощью регистра адреса легко обеспечить циклическое выполнение фрагментов кода микропрограмм.

СУАМ под управлением сигналов S1 и S0 обеспечивает два основных режима адресации микрокоманд: принудительный, когда адрес следующей микрокоманды задается в адресном поле текущей микрокоманды и подается на входы D3 – D0 или/и входы R3 – R0, и естественный, при котором адрес следующей микрокоманды определяется счетчиком микрокоманд. Управление СУАМ (выбор источника адреса следующей микрокоманды, задание режима адресации и пр.) осуществляется сигналами, кодируемыми специальными разрядами адресного поля микрокоманды. Для сокращения разрядности поля микрокоманды, управляющего работой БМУ, в микропроцессорном комплекте К1804 имеется контроллер последовательности микрокоманд К1804ВУ3. Контроллер представляет собой преобразователь «сильнокодированного» 4-разрядного поля инструкции в адресной части микрокоманды и признака ветвлений в наборы из восьми управляющих сигналов для узлов, входящих в БМУ. Преобразователь реализован в виде ПЗУ емкостью в 32 8-разрядных слова.

При объединении 4-разрядных секций СУАМ можно реализовать БМУ с произвольной разрядностью адреса микрокоманд, при этом диапазон адресуемых ячеек составляет до 2^{4n}, где n — число последовательно включенных секций СУАМ. При объединении секций необходимо соединить цепи переносов n секций СУАМ и объединить их общей шиной управления.

Функциональная схема $4n$-разрядной микропроцессорной системы, реализованной на БИС микропроцессорного комплекта К1804, показана на рис. 3.6.

Кратко поясним работу системы. Выбранная из микропрограммной памяти микрокоманда запоминается в регистре микрокоманд. С помощью кода из операционного поля регистра микрокоманд ЦПЭ К1804ВС1 обеспечивает выполнение операции, заданной в микрокоманде. На вход D3-D0 секции ЦПЭ данные могут поступать через мультиплексор из входной шины ВхД или поля D микрокоманды. Разрядность обрабатываемых данных $4n$ определяется числом n используемых секций ЦПЭ. После исполнения очередной микрокоманды блок микропрограммного управления реализует переход к следующей. Для сокращения разрядности поля микрокоманды, управляющего работой БМУ, в адресном поле текущей микрокоманды указывается только тип перехода, а набор сигналов для управления СУАМ К1804ВУ1 формируется с помощью контроллера последовательности микрокоманд К1804ВУ3. При формировании адреса при условных переходах в БМУ анализируется условие перехода, которым может являться один из признаков результата выполнения предыдущей микрооперации. Источниками прямого адреса СУАМ, кроме адреса перехода из регистра микрокоманд, могут быть вектор прерывания преобразователь начального адреса, не показанные на рис. 3.6. Разрядность формируемого СУАМ адреса и соответствующая ему емкость

микропрограммной памяти определяются числом k включенных секций СУАМ.

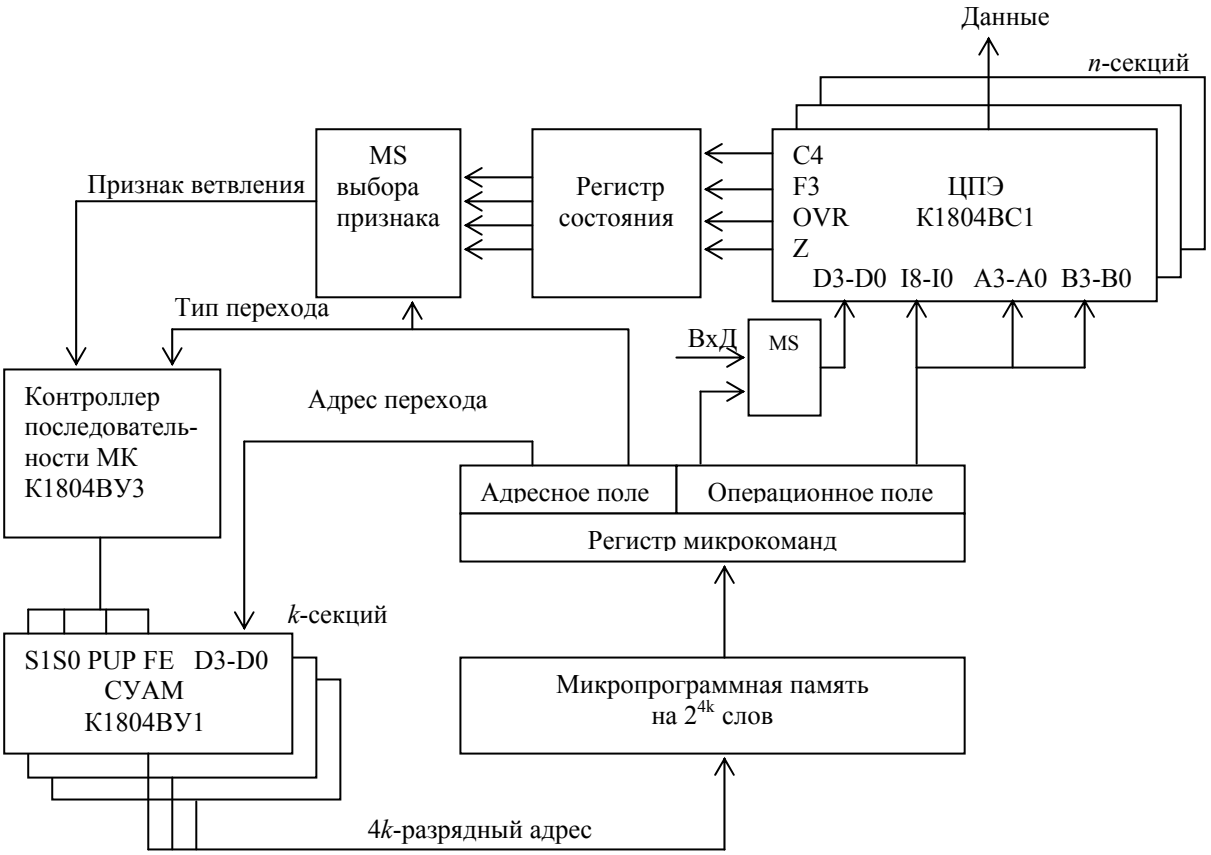


Рис. 3.6. Функциональная схема микропроцессорной системы

4. Методические указания к лабораторному практикуму

В лаборатории «Организация ЭВМ, микропроцессоры и микропроцессорные системы» изучаются и исследуются два основных класса микропроцессоров: микропроцессоры с фиксированными разрядностью и списком команд и микропроцессоры с наращиваемой разрядностью слова и микропрограммным управлением (секционные микропроцессоры). Исследования МП выполняются на лабораторных установках, реализованных на базе универсального лабораторного стенда DiLaB (Digital Laboratory Board).

Универсальные стенды DiLaB позволяют моделировать типовые структуры МП-систем, построенных на СБИС микропроцессорных комплектов различных микропроцессоров. Состав моделируемого ядра МП-системы определяется типом исследуемого МП и рассматривается ниже.

С помощью лабораторных стендов изучаются функциональные возможности и внутренняя структура МП различных типов, системы команд (микрокоманд) этих МП, принципы проектирования и отладки микропроцессорных систем на основе МП.

Стенд DiLaB (рис. 4.1) содержит программируемую пользователем вентильную матрицу FPGA (Field programmable gate array) – ПЛИС EP2C8F256-8 фирмы Altera), внешнее ОЗУ - SRAM емкостью 8 Кбайт, конфигурационная память - EPCS4 и энергонезависимую память данных - SPI flash 4 Mbit (в стенде не используется).

Рис. 4.1. Общий вид платы DiLaB

Ядро моделируемой МП-системы с исследуемым МП реализуется на плате расширения PB-CII (Cyclone II) с установленной вентильной матрицей FPGA, запрограммированной в виде IP-ядра соответствующего МП.

Для управления работой стенда на его панели управления размещены следующие элементы управления и индикации:

- 4 кнопки-клавиши PUSH BUTTENS (UP, DOWN, LEFT, RIGHT);
- 8 переключателей блока SWITCHES (SW1 – SW8);
- 16-кнопочная клавиатура (KEYPAD);
- 8 светодиодных индикаторов блока LED INDICATORS (LED1 – LED8);
- 4 семисегментных индикатора LED MONITOR (Digit1 – Digit4).

Назначение элементов управления и индикации лабораторного стенда зависит от моделируемой МП-системы с исследуемым микропроцессором.

В данном пособии представлены методические указания по проведению исследований микропроцессорных систем на основе однокристальных микропроцессоров КР580ВМ80А и секционных МП серии К1804.

4.1. Микропроцессорные системы на основе однокристальных МП

Цель данного цикла работ - ознакомление с организацией МП-систем на основе однокристальных МП 8080 (КР580ВМ80А) и приобретение навыков программирования микропроцессоров указанного типа.

Лабораторные работы выполняются на лабораторной установке DiLaB 8080, реализованной на базе универсального стенда DiLaB (рис. 4.1) с встроенной моделью ядра микропроцессорной системы с МП 8080.[1] При выполнении лабораторных работ исследуется МП-система, реализованная на плате расширения PB-CII (Cyclone II) с вентильной матрицей FPGA, которая запрограммирована в виде IP-ядра МП КР580ВМ80А [T80 cpu http://opencores.com/].

4.1.1. Структура и режимы работы лабораторного стенда DiLaB 8080

Внутренняя организация лабораторного стенда соответствует типовой структуре микропроцессорной системы, выполненной на СБИС микропроцессорного комплекта К580. Моделируемое ядро системы объединяет модель блока центрального процессора (ЦПЭ К580ВМ80А), модель генератора двухфазных тактовых последовательностей импульсов Ф1 и Ф2 (КР580ГФ24), модель системного контроллера сигналов управления обменом информации (КР580ВК38), модель блока шинных формирователей адреса и данных КР580ВА86, модель порта ввода-вывода параллельной информации КР580ВВ55 и блок оперативного запоминающего устройства. Функциональная схема МП-системы с МП 8080 (рис. 4.2) кроме перечисленных блоков содержит устройства ввода и вывода информации. Устройство ввода, реализованное в

[1] Лабораторная установка DiLaB 8080, реализованная на базе универсального стенда DiLaB в среде Quartus II, разработана бакалавром кафедры компьютерных систем и программных технологий Табачником И.Е.

виде набора клавишных переключателей, обеспечивает загрузку информации в память системы и используется для задания режимов работы стенда. Устройство вывода представлено набором светодиодных индикаторов и табло жидкокристаллических индикаторов (ЖКИ), отражающих состояние шин процессорного блока в различных режимах работы стенда.

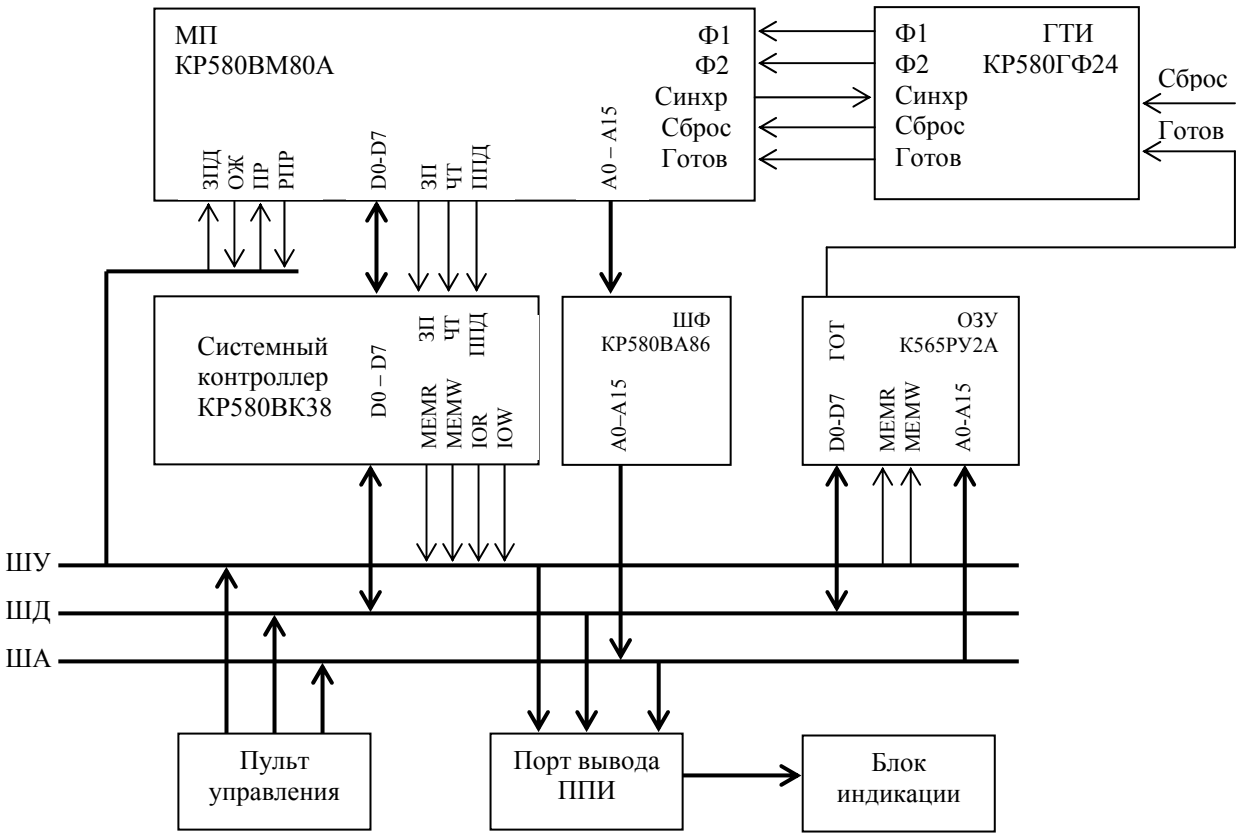


Рис. 4.2. Функциональная схема лабораторного стенда DiLaB 8080-2 с МП КР580ВМ80А

Лабораторный стенд DiLaB 8080 обеспечивает работу МП-системы в двух основных и нескольких дополнительных (вспомогательных) режимах.

Основными режимами являются режим выполнения программы WORK (РАБОТА - ВЫПОЛНЕНИЕ ПРОГРАММЫ) и режим доступа к ячейкам памяти системы DMA (direct memory access – прямой доступ к памяти). Основные режимы работы микротренажера в свою очередь имеют специальные подрежимы.

Для управления работой стенда DILAB 8080 на его панели управления размещены следующие элементы управления и индикации:

- 4 кнопки-клавиши PUSH BUTTENS (UP, DOWN, LEFT, RIGHT).

Клавиша LEFT обеспечивает переключение режимов работы: из режима WORK S в режим DMA и обратно при повторном нажатии.

Клавиша RIGHT («ПУСК») используется для формирования сигнала «ПУСК» при выполнении команд по машинным циклам в пошаговом режиме WORK S.

Клавиши UP и DOWN обеспечивают последовательный перебор адресов в режиме DMA. Кнопка UP (+) инкрементирует адрес, а кнопка DOWN (–) декрементирует его.

- 8 переключателей блока SWITCHES (SW1 – SW8) используются в режиме DMA для задания кода старшего и младшего байтов адреса, а также двоичного кода вводимых 8-битных данных;

- 16-кнопочная клавиатура (KEYPAD), отдельные кнопки которой (кнопки «A», «B», «C», «D», «*», «#», «0», «1») используются при формировании сигналов управления режимами работы стенда. Назначение кнопок клавиатуры KEYPAD рассматривается ниже;

- 8 светодиодных индикаторов блока LED INDICATORS (LED1 – LED8) в режиме пошагового выполнения WORK S индицируют значения сигналов байта состояния текущего машинного цикла. В других режимах работы микротренажера индикаторы LED1 – LED8 отключены;

- 4 семисегментных индикатора LED MONITOR (Digit1 – Digit4) используются для вывода результатов выполнения пользовательской программы блоками статической и динамической индикации. Особенности работы микротренажера в режиме ВЫВОДА параллельной информации рассматриваются в п. 4.1.4.

Название текущего режима работы микропроцессорной системы - WORK или DMA высвечивается в левом углу верхней строки двухстрочного табло LCD MONITOR

Полный перечень элементов управления режимами работы лабораторного стенда DiLaB 8080 приведен в приложении 3.

Основным подрежимом выполнения программы является режим пошагового выполнения команд по машинным циклам (с остановкой после каждого машинного цикла) – **режим WORK S M** (S – step, M – машинный цикл). В этом режиме запуск на выполнение машинного цикла осуществляется при нажатии клавиши RIGHT (ПУСК) блока PUSH BUTTONS. Переход в режим WORK S M выполняется автоматически при включении питания, при нажатии кнопки «*» - RESET, клавиши LEFT (при переходе из режима DMA в режим WORK) или после выполнения последней команды программы HLT. В режиме WORK S M на экране дисплея LCD MONITOR высвечивается информация о ячейке памяти, к которой происходит обращение (ее адрес и содержимое), а также надпись WORK S M (рис. 4.3).

W	O	R	K	S	M			D7	D6	D5	D4	D3	D2	D1	D0

Данные адресуемой ячейки памяти

A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0

Адрес ячейки памяти

Рис. 4.3. Формат информации на экране дисплея LCD MONITOR в пошаговом режиме выполнения программы по машинным циклам

При выполнении команд программы в пошаговом режиме WORK S M по машинным циклам светодиоды LED0 – LED7 блока LED (рис. 4.4) отображают значения сигналов байта состояния, который МП формирует в первом такте машинного цикла. Отдельные биты байта состояния являются управляющими сигналами. Используя эти сигналы и выходные сигналы микропроцессора WR и DBIN, системный контроллер формирует сигналы «Чтение» и «Запись» отдельно для памяти и отдельно для периферийных устройств ввода/вывода. При реализации всего списка команд используется 10 видов машинных циклов: типы машинных циклов и соответствующие им сигналы байта состояния приведены в приложении 2.

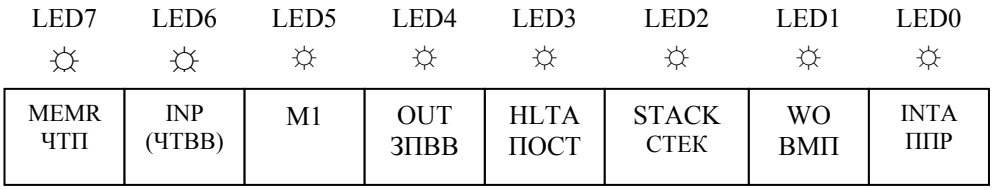


Рис. 4.4. Сигналы байта состояния микропроцессора 8080

Выполнение команд в пошаговом режиме контролируется в каждом машинном цикле по значениям сигналов ШИНЫ АДРЕСА и ШИНЫ ДАННЫХ на двухстрочном табло LCD MONITOR и сигналам байта состояния на индикаторах LED0 – LED7 блока LED. В частности, при считывании кода операции команды (цикл выборки команды M1) важно убедиться в наличии сигналов M1 (LED5) и ЧТП (LED7) (проверяется по одноименным индикаторам байта состояния).

В режиме WORK S M клавиши UP, DOWN блока PUSH BUTTONS, кнопки A, B, D, # клавиатуры KEYPAD и переключатели SWITCHES (SW1 – SW8) деактивируются (отключаются). Клавиши LEFT и RIGHT блока PUSH BUTTONS и кнопки «*», «C» и «1» клавиатуры KEYPAD активны.

При изучении особенностей функционирования МП и отладке программ команды программы следует выполнять в пошаговом режиме по машинным циклам.

Режим WORK S C (S – step, C – command) является режимом пошагового выполнения команд по командным циклам (с остановкой после каждого командного цикла). Перейти в этот режим можно только из режима WORK S M при нажатии кнопки «1» клавиатуры KEYPAD. В режиме WORK S C при нажатии клавиши RIGHT (ПУСК) блока PUSH BUTTONS выполняется очередная команда программы, при этом на экране дисплея LCD MONITOR высвечивается код операции и адрес следующей команды, а также надпись WORK S C (рис. 4.5) . Режим WORK S C удобен для «быстрого» просмотра кодов операций команд с одновременным их исполнением.

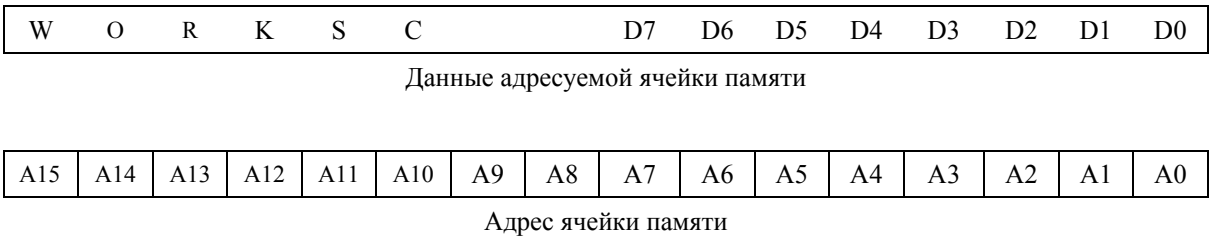


Рис. 4.5. Формат информации на экране дисплея LCD MONITOR в пошаговом режиме выполнения программы по командным циклам

Режим WORK C (C - continues) - режим автоматического (непрерывного) выполнения команд программы. Режим WORK C устанавливается при нажатии кнопки «C» клавиатуры KEYPAD. В этом режиме программа выполняется полностью, (если это не циклическая программа). По завершению программы (после выполнения последней команды HLT в любом режиме WORK) процессор переводится в состояние «ОСТАНОВА», в котором выходные каскады разрядов шин адреса и данных МП находятся в состоянии высокого сопротивления. На экране дисплея LCD MONITOR высвечивается сообщение о завершении программы (рис. 4.6). Выход из состояния ОСТАНОВА (перевод микротренажера в один из основных режимов работы) выполняется либо при нажатии клавиши LEFT (в этом случае устанавливается режим работы DMA, обеспечивающий возможность проконтролировать результат выполнения программы, зафиксированный в памяти), либо при нажатии кнопки «*» для перевода в режим WORK S.

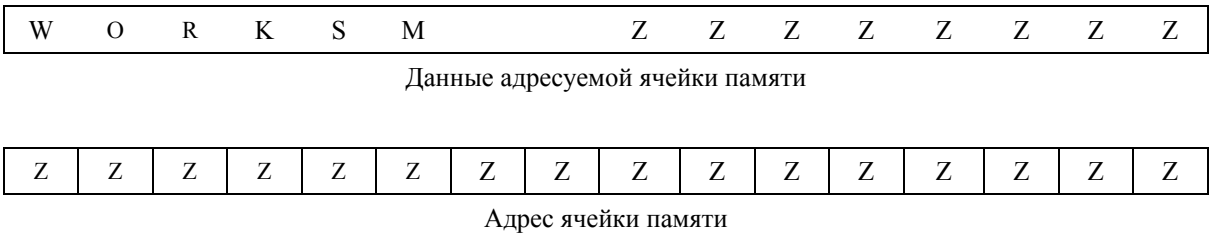


Рис. 4.6. Формат информации на экране дисплея LCD MONITOR в режиме ОСТАНОВА

При выполнении «некорректных» циклических программ (без «ОСТАНОВА») в режиме WORK C на экране дисплея LCD MONITOR высвечивается надпись «WORK C» (рис. 4.7).

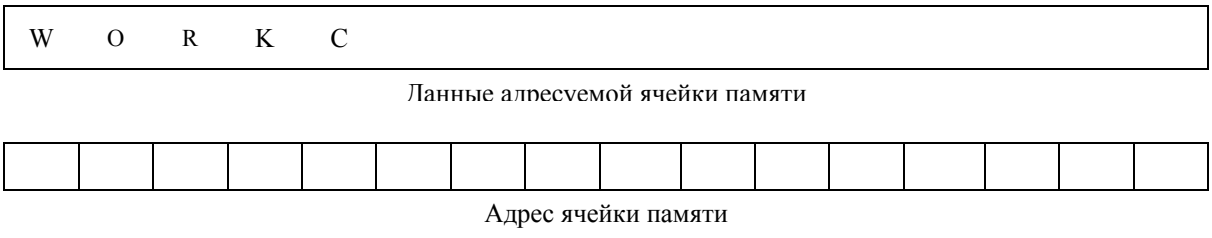


Рис. 4.7. Формат информации на экране дисплея LCD MONITOR в режиме WORK C

Перейти в режим непрерывного выполнения команд WORK C можно только из режима WORK S (в остальных режимах работы системы кнопка «C» деактивирована).

Режим OUT. Специальным подрежимом работы стенда DILAB 8080 является режим вывода параллельной информации OUT (ВЫВОД). В режиме OUT для индикации результатов выполнения программы используется блок семисегментных индикаторов LED MONITOR, который может быть реализован в виде блоков статической или динамической индикации. Особенности работы микротренажера в режиме ВЫВОДА рассматриваются в п.4.1.4 одновременно с рассмотрением работы блоков статической и динамической индикации. В режим OUT микротренажер переводится из режима WORK при нажатии кнопки «0» клавиатуры KEYPAD, при этом устанавливается режим отображения результатов блоком динамической индикации (рис. 4.8). Для перехода в режим отображения блоком статической индикации (рис. 4.9) необходимо нажать кнопку «#» клавиатуры KEYPAD. Выполнение команд в режиме OUT может осуществляться либо в пошаговом режиме, либо в режиме непрерывной работы. Исполняемый режим OUT идентифицируется надписью на экране дисплея LCD MONITOR: OUTS (статическая индикация) или OUTD (динамическая индикация) с добавлением буквы названия режима исполнения команд (S – по шагам, C – непрерывное исполнение).

O	U	T	D	S				D7	D6	D5	D4	D3	D2	D1	D0

Данные адресуемой ячейки памяти

A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0

Адрес ячейки памяти

Рис. 4.8. Формат информации на экране дисплея LCD MONITOR об адресе и данных в режиме вывода данных на блок динамической (D) индикации при пошаговом исполнении команд (S)

O	U	T	S	S			D	D7	D6	D5	D4	D3	D2	D1	D0

Данные адресуемой ячейки памяти

A	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0

Адрес ячейки памяти

Рис. 4.9. Формат информации на экране дисплея LCD MONITOR об адресе и данных в режиме вывода данных на блок статической (S) индикации при пошаговом исполнении команд (S)

В режимах непрерывного выполнения команд формат отображаемой информации изменится (рис. 4.10 и рис. 4.11).

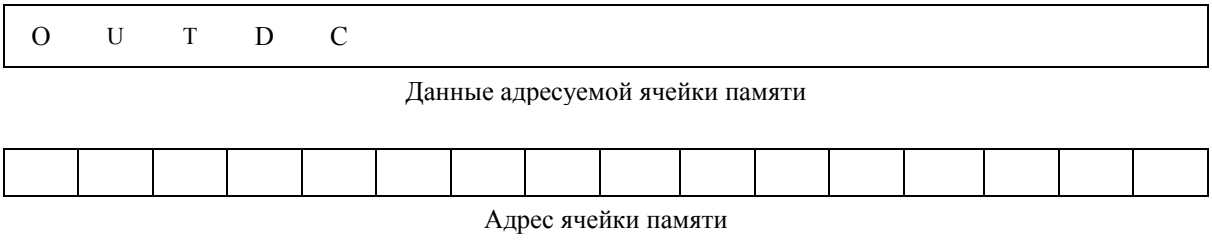


Рис. 4.10. Формат информации на экране дисплея LCD MONITOR в режиме ВЫВОДА данных на блок динамической (D) индикации при непрерывном исполнении команд (C)

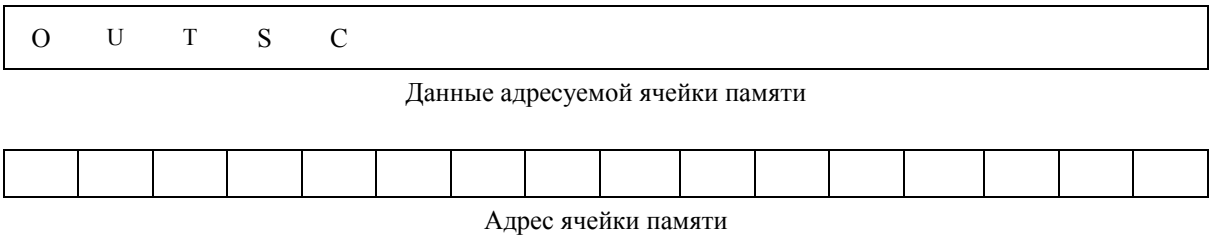


Рис. 4.11. Формат информации на экране дисплея LCD MONITOR в режиме ВЫВОДА данных на блок статической (S) индикации при непрерывном исполнении команд (C)

Выход из режима ВЫВОД реализуется при нажатии кнопки RESET («*») или повторном нажатии кнопки «0».

Режим DMA контроля и модификации содержимого памяти МП-системы

В режиме доступа к ячейкам памяти обеспечивается возможность чтения и при необходимости модификация содержимого адресуемых ячеек памяти. Этот режим работы стенда, также называемый режимом прямого доступа в память DMA, соответствует одноименному режиму работы МП КР580ВМ80.

В стенде DiLaB 8080 основной режим доступа к ячейкам памяти DMA, в котором пользователь может задать произвольные адреса ячеек и модифицировать их содержимое, дополнен подрежимом доступа к последовательно адресуемым ячейкам памяти. В этом подрежиме с помощью специальных кнопок +/- (клавиши «UP» и «DOWN» блока PUSH BUTTONS) осуществляется выбор последовательно адресуемых ячеек памяти с возможностью модификации их содержимого.

Режим доступа к ячейкам памяти идентифицируется надписью **DMA** в левом углу верхней строки двухстрочного табло LCD MONITOR. На табло также отображается содержимое адресуемых ячеек памяти. 16-битный адрес ячейки памяти индицируется в нижней строке табло ЖКИ, а данные (содержимое ячейки памяти) - в верхней строке. Три старших разряда 16-битного адреса (из-за ограниченного объема ОЗУ стенда) имеют значение «0» (см. рис. 4.12).

D	M	A						D7	D6	D5	D4	D3	D2	D1	D0

Данные адресуемой ячейки памяти

0	0	0	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0

Адрес ячейки памяти

Рис. 4.12. Формат информации на экране дисплея LCD MONITOR об адресе и данных в режиме доступа к ячейкам памяти с произвольными адресами DMA

Доступ к ячейкам памяти с произвольными адресами обычно применяют для коррекции пользовательской программы, когда в нее необходимо внести локальные изменения кодов некоторых команд, а также при чтении или модификации данных в отдельных ячейках памяти.

Для упрощения процедуры задания адреса при переходе в режим DMA по умолчанию старший байт адреса (AH) принимается равным «0» и при задании адреса ячейки можно вводить только младший байт адреса (AL).

Двоичный код значения вводимого операнда (старшего или младшего байтов адреса A и непосредственные 8-битные данные D7 – D0) задается с помощью восьми переключателей SWITCHES (SW1 – SW8). Вводимое значение операнда **фиксируется** в соответствующем буферном регистре (адреса или данных) при нажатии специально выделенных кнопок 16-кнопочной клавиатуры KEYPAD: кнопки «**A**» - **младший байт адреса**, кнопки «**B**» - **старший байт адреса**, кнопки «**D**» - непосредственные **8-битные данные**.

Запись данных в память (кодов команд и собственно данных) осуществляется побайтно. Для записи данных необходимо:

- перевести стенд в режим DMA;
- задать двухбайтный адрес ячейки памяти. Задание адреса выполняется в два этапа, при этом порядок задания байтов адреса значения не имеет. Для задания адреса с помощью переключателей Switches набирается 8-разрядный код вводимого байта, после чего нажимается кнопка «A» (при вводе младшего байта адреса) или кнопка «B» (при вводе старшего байта адреса) 16-кнопочной клавиатуры KEYPAD. В результате соответствующий байт 16-битного адреса фиксируется в буферном регистре адреса и на табло ЖКИ-монитора высвечивается адрес ячейки памяти и ее содержимое. Если старший байт адреса не требует коррекции, его (старший байт) можно не вводить. После задания адреса содержимое адресуемой ячейки сразу же отображается в поле «данные» табло LCD MONITOR;
- для модификации содержимого адресуемой ячейки требуемый код данных набирается на переключателях SWITCHES (SW1 – SW8), после чего нажимается кнопка «D» («запись данных») 16-кнопочной клавиатуры KEYPAD. Правильность ввода данных контролируется по табло LCD MONITOR.

В режиме DMA при кратковременном нажатии клавиш «UP» или «DOWN» осуществляется выбор соседней ячейки памяти соответственно с большим или меньшим адресом. Последовательное нажатие/отпускание клавиши «UP» или «DOWN» обеспечивает возможность доступа к ячейкам памяти по последовательным адресам. Этот подрежим доступа к ячейкам памяти удобно использовать для записи исходного текста пользовательской программы в программную память и просмотра содержимого последовательно адресуемых ячеек памяти.

Подготовка системы к работе осуществляется сигналом «RESET» (НУ), который формируется при нажатии кнопки «*» клавиатуры KEYPAD. При поступлении сигнала НУ счетчик команд, триггеры разрешения прерывания и подтверждения захвата МП сбрасываются в «0», микротренажер переводится в режим пошагового выполнения WORK S, процессор ожидает поступления сигнала RIGHT (ПУСК) для выполнения команды, записанной по нулевому адресу.

Микротренажер допускает возможность выполнения команд программы, размещенных по произвольным адресам памяти системы. Если начальная установка не выполняется (не нажимается кнопка «*» RESET), то при переходе из режима DMA в режим WORK S процессор начинает выполнение программы с команды, размещенной по адресу, который был задан последним в режиме DMA.

Сводная информация о назначении элементов управления режимами работы тренажера представлена в приложении 3.

Программирование пользовательских задач на учебных стендах выполняется методом «ручного» программирования - программирования в машинных кодах. При программировании в машинных кодах пользователь максимально приближен к аппаратным средствам. Он должен хорошо знать структуру процессора, его систему команд и другие особенности архитектуры. «Ручное» программирование в наибольшей степени подходит для целей обучения основам организации ВМ, принципам построения и функционирования.

При программировании в машинных кодах все элементы прикладной программы (коды операций, адреса, данные) представляются в двоичном формате или в эквивалентных, но более удобных для восприятия и отображения восьмеричном или шестнадцатеричном форматах. К сожалению, при таком способе программировании приходится сталкиваться с рядом трудностей. В частности, необходимо помнить коды операций отдельных команд, входящих в состав системы команд микропроцессора, трудно следить за абсолютными адресами памяти, особенно в программах с большим числом условных переходов, сложно модифицировать разработанную программу, вставляя в нее новую команду или группу команд. Поэтому непосредственное программирование в машинных кодах применяется редко.

На практике более распространено «ручное» программирование на языке ассемблере, позволяющем в значительной степени устранить недостатки программирования на машинном языке. Язык ассемблер допускает представление всех элементов программы в символьной (буквенно-цифровой) форме, отражающей их содержательный смысл. Преобразование символических имен в машинный код при ручном ассемблировании осуществляет сам программист (при автоматическом программировании подобное преобразование осуществляется с помощью специальной программы, также называемой ассемблером). Ассемблирование вручную упрощает написание программ, позволяет уменьшить количество ошибок кодирования, систематизирует документирование программ. При ручном ассемблировании программу рекомендуется оформлять в виде последовательности строк следующего формата:

Адрес Код операции Мнемоника команды Комментарий

При таком представлении программы каждая строка отображает содержимое одной ячейки программной памяти. Написание программы при ручном ассемблировании не требует выполнения жестких правил синтаксиса автоматических ассемблеров.

Учебные программы, выполняемые в лаборатории на микротренажерах, моделирующих работу микропроцессорной системы, обычно разрабатываются на ассемблере, а их кодирование осуществляется вручную.

4.1.2. Программа работы на стенде при изучении МПК серии К580

Работа рассчитана на несколько посещений. Ее содержанием является:

– ознакомление с внутренней организацией МП КР580ВМ80 и режимами его работы;

– детальное изучение системы команд МП 8080 и способов адресации;

– изучение программируемого интерфейса ввода-вывода;

– приобретение навыков программирования и построения простейших контроллеров на основе МПК серии К580.

Для выполнения заданий этого раздела, прежде всего, необходимо приобрести навыки управления стендом в различных режимах его работы. Запись данных в ОЗУ и чтение содержимого памяти осуществляется в режиме DMA в соответствии с последовательностью действий, рассмотренных выше.

Переключите стенд в режим DMA (нажмите клавишу Left блока PUSH BUTTENS). С помощью блока переключателей SWITCHES и кнопок A и B клавиатуры задавайте двухбайтные адреса произвольных ячеек памяти и по двухстрочному табло LCD MONITOR считывайте содержимое адресуемых ячеек, контролируя при этом адреса ячеек.

Для записи данных в ячейку памяти задайте двухбайтный адрес ячейки, с помощью блока переключателей SWITCHES наберите 8-разрядный код вводимых данных, после чего нажмите кнопку D клавиатуры. Правильность

ввода данных контролируется по табло LCD MONITOR. При необходимости скорректируйте данные (повторно запишите исходные данные по заданным адресам).

Для иллюстрации режима записи данных в ОЗУ запишите требуемые восьмеричные числа по заданным адресам.

200	по адресу	H=001	L=377
201	по адресу	H=002	L=000
202	по адресу	H=002	L=001

ПРИМЕЧАНИЕ: в этом задании и во всех последующих программах адреса, команды и данные приводятся в восьмеричной системе счисления. Для представления 16-битного адреса памяти используются два байта. Старший байт адреса обозначается H, а младший – L. Если адрес H не указан, то он соответствует восьмеричному коду 000.

Изучение режимов работы МП КР580ВМ80 и приобретение начальных навыков программирования устройств на его основе выполняется с помощью десяти тестовых программ, тексты которых приводятся ниже.

Программа 1 позволяет загрузить требуемые данные в два заданных регистра РОН.

Загрузка данных в регистры процессора К580 в принципе не представляет труда и выполняется командами непосредственной загрузки *MVI Rd,data* и *LXI RP,data16*. Сложнее реализовать контроль правильности загрузки, поскольку из-за ограниченного числа выводов МП КР580ВМ80 нельзя непосредственно проверить содержимое регистров блока РОН. Существуют несколько косвенных способов контроля содержимого РОН. Например, можно переслать содержимое регистров в память и затем в режиме DMA проверить содержимое соответствующих ячеек памяти. В программе 1 использован именно этот способ чтения содержимого РОН (пересылка содержимого регистров в память выполняется командами с адресами 006 и 010).

Программа 1

Адрес	Код команды	Мнемоника команд	Комментарии
L=000	041	LXI H	;Загрузка начального адреса памяти
001	200	B2	;Мл. байт адреса
002	000	B3	;Ст. байт адреса
003	001	LXIB	;Запись исходных данных в регистры B и C
004	001	B2	;B2 → (C)
005	010	B3	;B3 → (B)
006	160	MOV M, B	;Запоминание (B)
007	043	INX H	;Формирование следующего адреса памяти
010	161	MOV M, C	;Запоминание (C)
011	166	HLT	;Останов

Проанализируйте и выполните программу 1 в пошаговом режиме по машинным циклам. Обратите внимание, что при выполнении программы после считывания команд, требующих для своей реализации дополнительного цикла внешнего обмена, порядок высвечиваемых адресов памяти нарушается. В этих командах при выполнении дополнительных циклов обмена на двухстрочном

табло LCD MONITOR высвечивается адрес ячейки памяти, к которой происходит обращение, а на индикаторах ДАННЫЕ отображается записываемая (считываемая) информация.

В режиме DMA проверьте содержимое ячеек памяти с адресами L=200 и L=201 (H=000). Фактическое содержимое ячеек памяти сравните с исходными данными, вводимыми в заданные регистры.

Проанализируйте результат выполнения команды HLT. При ее выполнении выходы буферных регистров АДРЕС и ДАННЫЕ переводятся в состояние высокого сопротивления (на табло LCD MONITOR разряды шин адреса и данных принимают значение *Z*), в байте состояния формируются сигнал ПОСТ, свидетельствующий о переходе МП в состояние «Останов». Выполнение программы прекращается. В состоянии «Останова» микропроцессор может находиться в течение любого временного интервала.

Разработайте *программу 2*, иллюстрирующую работу процессора при выполнении команд обращения к стеку.

В микропроцессорных системах с МП К580 стек моделируется в памяти. Для определения области ОЗУ, выделяемой под стек, необходимо использовать команду загрузки указателя стека *LXI SP data16*. Обращение к стеку выполняется командами *PUSH RP* (загрузить в стек содержимое регистровой пары, указываемой в команде) и *POP RP* (извлечь из стека).

В качестве примера разработайте программу, позволяющую извлечь данные, предварительно записанные в стек, изменить их на единицу (используя операцию инкремента - декремента) и занести скорректированные данные снова в стек. Для определенности до начала выполнения программы запишите в стек (ячейки памяти с адресами L=200 и L=201) числа 222 и 333 соответственно.

Один из возможных алгоритмов программы 2 реализует следующую последовательность действий:

- инициализировать указатель стека для определения области ОЗУ, выделяемой под стек, например, значением SP_H=000 и SP_L=200;
- извлечь из стека данные, предварительно загруженные в него, и загрузить их в адресуемую регистровую пару (варианты по указанию преподавателя: регистровые пары B, D, H);
- модифицировать данные в регистрах путем увеличения содержимого одного регистра и уменьшения содержимого другого регистра пары на 1;
- запомнить модифицированные данные в стеке;
- завершить выполнение программы командой Останов.

Выполните программу 2 в пошаговом режиме. Обратите внимание на то, что при выполнении команд *PUSH RP* и *POP RP* при последовательных нажатиях кнопки RIGHT (ПУСК) после считывания кода команды на табло LCD MONITOR высвечивается результат выполнения этих команд, т.е. содержимое, загружаемое (извлекаемое) в (из) ячейки памяти, адресуемой указателем стека. После выполнения программы прочтите содержимое ячеек с адресами L=200 и L=201. Объясните полученный результат и ответьте на вопросы:

Почему при работе со стеком необходимо инициализировать указатель SP?

Почему при выполнении команд обращения к стеку содержимое указателя стека изменяется на два?

В каком направлении "растет" стек при загрузке?

Почему при выполнении команд PUSH RP загружаются ячейки памяти с адресами (SP-1) и (SP - 2), а при выполнении команды POP RP информация извлекается из ячеек памяти с адресами (SP) и (SP +1)?

Программа 3 реализует суммирование содержимого аккумулятора с содержимым ячейки памяти.

Программа 3

Адрес	Код команды	Мнемоника команд	Комментарии
L=000 001 002	041 200 000	LXI H B2 B3	;Загрузка начального адреса памяти ;Мл. байт адреса ;Ст. байт адреса
003 004	076 010	MVI A B2	
005	206	ADD M	
006	167	MOV M, A	
007	166	HLT	

Выполните программу 3 в пошаговом режиме и проверьте содержимое ячейки с адресом L=200. Повторите выполнение программы в автоматическом режиме. Для этого необходимо с помощью кнопки «*» вывести МП из режима «Останов» и включить режим непрерывного выполнения (нажать кнопку «С»). После каждого завершения программы проверяйте содержимое ячейки памяти с адресом L=200 и фиксируйте результат.

Разработайте и выполните программу 3.1 вычитания двух чисел, одно из которых расположено в регистре блока РОН (Варианты: регистры B, C, D, E, H, L). Результат выполнения сохраняйте в памяти.

Разработайте и выполните программу 3.2 сложения двоично-десятичных чисел. Рассмотрите действие команды DAA для различных значений суммируемых операндов. Результат выполнения фиксируйте в памяти.

Разработайте и выполните программу 3.3 вычитания двоично-десятичных чисел. При написании этой программы учтите, что команда DAA не корректирует результат вычитания двоично-десятичных чисел. Полезно напомнить, что операция вычитания эквивалентна операции сложения в дополнительных кодах. Дополнительный код положительного числа есть само число, а дополнительный код отрицательного числа есть число, являющееся дополнением до максимально возможного числа заданного числа разрядов используемой системы исчисления плюс единица. В соответствии с этим определением дополнительный код двухразрядного двоичного числа $N_{доп}$ определяется по правилу $N_{доп} = (99 - N) + 1$. При вычислении $N_{доп}$ можно

использовать обычное двоичное вычитание, поскольку заем из старшего десятичного разряда не возникает. Число $N_{доп}$, вычисленное по этому правилу есть обычное двоично-десятичное число. Используя алгоритм сложения двоично-десятичных чисел, получить результат вычитания двоично-десятичных чисел не представляет труда.

Разработайте и выполните программу 3.4 сложения двух 16-разрядных двоичных чисел. Объясните назначение команды *ADC* R_S. Операнды команды разместите в регистровых парах B и D, а результат выполнения загрузите в память с помощью команд, прямо адресующих приемник.

Разработайте и выполните программу 3.5 вычитания двух 16-разрядных двоичных чисел. Объясните назначение команды *SBB* R_S. Операнды команды разместите в регистровых парах B и D, а результат выполнения загрузите в память с помощью команд, косвенно адресующих приемник.

Программа 4 реализует «обнуления» заданной области памяти.

Программа «обнуления» представлена циклическим алгоритмом. Перед выполнением программы проверьте содержимое памяти с адресами от L=011 до L=040. В указанных ячейках могут присутствовать ненулевые данные. Почему?

Программа 4

Адрес	Код команды	Мнемоника команд	Комментарии
L=000	227	SUB A	;Обнуление A
001	041	LXI H	
002	011	B2	
003	000	B3	
004	167	MOV M, A	
005	043	INX H	
006	303	JMP	
007	004	B2	
010	000	B3	

Выполните программу 4 в автоматическом режиме (для переключения в режим непрерывного выполнения нажмите кнопку «C»). После выполнения программы перейдите в режим DMA и проконтролируйте содержимое ячеек по адресам от L=011 до L=040. Повторите выполнение программы в пошаговом режиме после сброса. Убедитесь, что часть программы «стерта». Почему? Запишите в отчете оставшуюся часть программы. Какова роль стертой части программы?

Отметим, что рассмотренная программа представляет собой пример некорректной программы, так как в ней не предусмотрены средства останова. Какую группу команд необходимо добавить в программу для корректного завершения программы?

Программа 5 иллюстрирует формирование Слова Состояния Программы PSW при выполнении различных групп команд микропроцессора.

Программа 5

Адрес	Код команды	Мнемоника команд	Комментарии
L=000	006	MVI B	
001	200	B2	
002	076	MVI A	
003	001	B2	
004	250	XRA B	
005	027	RAL	
006	061	LXI SP	
007	202	B2	
010	000	B3	
011	365	PUSH PSW	
012	166	HLT	

Проанализируйте значение Слова Состояния Программы PSW, формируемое при выполнении команд программы. Результат анализа представьте в виде:

Мнемоника команды	Значение PSW	
MVI B data	(A):= xxxxxxxx	РФ:= xx0x0x1x
MVI A data	(A):= 00000001	РФ:= x0x0x1x
XRA B	(A):= 10000001	РФ:= 10000110
и т.д.		

здесь x указывает на неопределенное значение бита в анализируемом байте PSW (при сбросе состояние PSW не определено); РФ – содержимое регистра флагов (признаков), кодируемое в соответствии с рис.2.4.

Выполните программу 5. После выполнения программы последовательно прочтите содержимое ячеек с адресами L=200 и L=201 и объясните полученный результат.

Программа 6 позволяет проверить действие команд условного перехода.

Перед выполнением программы запишите в память по адресу L=200 код 002, соответствующий нулевому значению всех флагов в регистре признаков, а по адресу L=004 - код 302, идентифицирующий команду условного перехода по ненулевому результату JNZ. Выполните программу в пошаговом режиме. Объясните, почему при заданном состоянии регистра признаков и выбранном типе команды условного перехода осуществляется циклическое выполнение программы.

Программа 6

Адрес	Код команды	Мнемоника команд	Комментарии
L=000 001 002	061 200 000	LXI SP B2 B3	;Инициализация указателя стека
003	361	POP PSW	;Извлечение из стека Слова Состояния Программы
004	*	*	;Код одной из восьми команд условных переходов ;(JNZ, JZ, JNC, JC, JPO, JPE, JP или JM)
005 006	000 000	B2 B3	;Младший байт адреса перехода ;Старший байт адреса перехода
007	166	HLT	
. . .			
200	**	-	;Слово состояния программы PSW,
201	000	-	;хранящееся в стеке

Измените данные в ячейке по адресу L=200 на число 102. Обратите внимание, как при этом изменится содержимое регистров признаков в слове PSW, которое хранится в стеке. Выполните программу и объясните, в чем разница по сравнению с предыдущей программой.

Выполните программу, изменив содержимое ячейки с адресом L=004 на число 312, и объясните, почему программа выполняется именно таким образом.

Проверьте действия команд условных переходов для каждого из четырех контролируемых признаков результата (нуль, перенос, четность, знак). Для этого повторите выполнение программы, изменяя код команды условного перехода и содержимое регистра признаков путем предварительной загрузки данных в ячейки памяти по адресам 200 и 204 в соответствии с табл.4.1.

Таблица 4.1

Данные по адресу L=004 (коды команд условного перехода)	Данные по адресу L=200 (содержимое регистра признаков)
322 и 332	002 и 003
342 и 352	002 и 006
362 и 372	002 и 202

Самостоятельно разработайте **п*рограмму* 7**, демонстрирующую действие команд вызова подпрограммы (CALL addr) и возврата в основную программу (RET). Используйте для этого команды LXI SP,data16, CALL addr и RET. Выполните программу в пошаговом режиме. Сравните команды CALL addr и JMP addr. В чем отличие этих команд при их исполнении?

Программа 8 поясняет действие команд RST N и RET при работе с прерывающими программами.

Программа 8

Адрес	Код команды	Мнемоника команд	Комментарии
L=000	061	LXI SP	Инициализация указателя стека
001	067	B2	
002	000	B3	
003	317	RST 1	Вызов подпрограммы по адресу L=010
004	166	HLT	
010	311	RET	

Выполните программу 8 в пошаговом режиме. Приведите полное описание работы программы по машинным циклам в виде следующей таблицы

Адрес (состояние ША)	Содержимое памяти (состояние ШД)	Комментарий
000	061	
001	067	
и т.д.		

Сравните команды CALL addr и RST N. В чем отличие в действии этих команд?

Программа 9 демонстрирует работу нескольких прерывающих друг друга вложенных программа.

Программа 9

Адреса	Код команды	Мнемоника команд	Комментарии
L=000	061	LXI SP	Инициализация указателя стека
001	067	B2	
002	000	B3	
003	317	RST 1	Вызов подпрограммы 1 по адресу L=010
004	166	HLT	
010	327	RST 2	Вызов подпрограммы 2 по адресу L=020
011	311	RET	Возврат из подпрограммы 1
020	337	RST 3	Вызов подпрограммы 3 по адресу L=030
021	311	RET	Возврат из подпрограммы 2
030	347	RST 4	Вызов подпрограммы 4 по адресу L=040
031	311	RET	Возврат из подпрограммы 3
040	357	RST 5	Вызов подпрограммы 5 по адресу L=050
041	311	RET	Возврат из подпрограммы 4
050	311	RET	Возврат из подпрограммы 5

Выполните программу 9 в пошаговом режиме. Проанализируйте ход ее выполнения и представьте алгоритм работы. Как изменится ход выполнения программы, если в вызываемой (прерывающей) программе будет отсутствовать команда RET?

4.1.3. Разработка программ индивидуальных заданий

По указанию преподавателя составьте и выполните программу индивидуального задания из списка заданий (приложение 6).

4.1.4. Знакомство с простейшими техническими средствами организации интерфейса микропроцессорных систем

В большинстве микропроцессорных систем обмен информацией между МП и периферийными устройствами осуществляется через специальные схемы сопряжения (интерфейсные схемы), выполненные на основе ИС малой и средней степени интеграции или в виде специальных БИС. В простейшем случае функции устройства сопряжения между МП и устройством ввода/вывода реализуются с помощью регистра, часто называемого портом.

В общем случае в качестве буферного устройства сопряжения между МП и устройствами ввода/вывода используют БИС программируемого параллельного интерфейса КР580ВВ55 (ППИ), входящую в состав микропроцессорного комплекта К580. БИС КР580ВВ55 представляет собой набор из трех 8-разрядных двунаправленных портов (каналов) A, B и C, предназначенных для подключения периферийных устройств к системным шинам (рис.4.13).

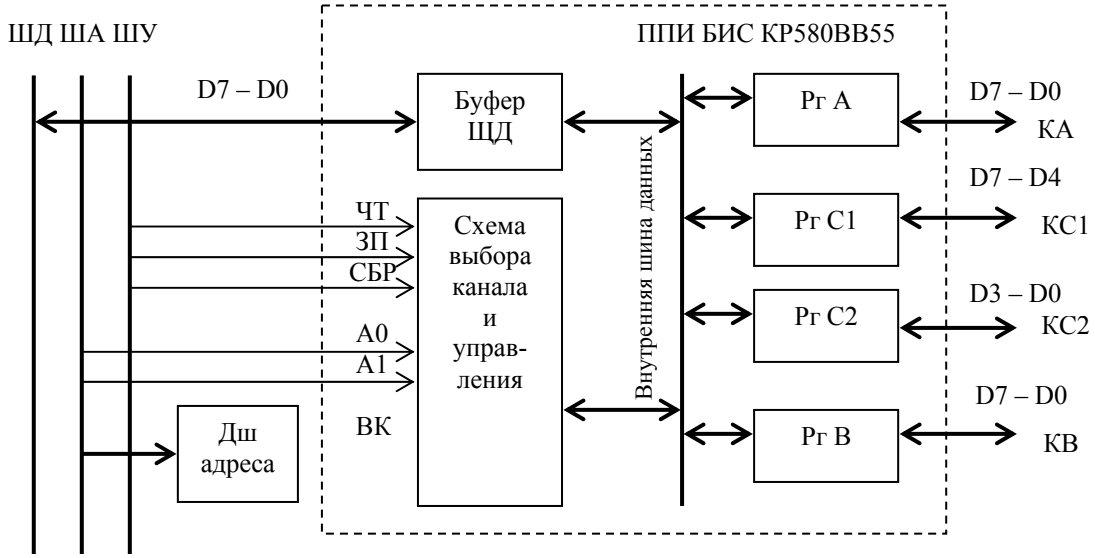


Рис.4.13. Структурная схема ППИ БИС

Структурная схема ППИ включает в себя двунаправленный буфер ШД, три информационных регистра каналов A, B, C, схему выбора канала и управления, которая содержит регистр управляющего слова. Связь БИС с системной ШД осуществляется через тристабильный 8-разрядный буфер ШД. Входы A0 и A1 подключаются к одноименным линиям шины адреса и используются для выбора внутреннего регистра ППИ (табл.4.2).

Таблица 4.2

Сигнал на входах		Адресуемый регистр
A1	A0	
0	0	Регистр канала A
0	1	Регистр канала B
1	0	Регистр канала C
1	1	Регистр управления

Старшие разряды шины адреса через дополнительный внешний дешифратор соединяются с входом ВК (выбор кристалла). Вход СБРОС используется для подключения одноименного сигнала, устанавливающего БИС К580ВВ55 в исходное состояние, при котором регистр управляющего слова обнуляется, а все порты (каналы A, B, C) переводятся в режим вывода. На входы ЧТ и ЗАП поступают управляющие сигналы, формируемые системным контроллером при выполнении команд IN port и OUT port. Эти сигналы обеспечивают считывание или запись информации в адресуемый порт.

Режим работы каналов программируется с помощью записываемого в регистр управления управляющего слова (рис.4.14).

СБИС ППИ реализует три режима обмена:

- режим 0 - основной режим ввода-вывода, позволяющий организовать синхронный обмен информацией между МП и внешним устройством по трем каналам, т.е. через регистры A, B и C. Вывод информации осуществляется по команде OUT port с фиксацией выводимой информации в регистрах канала. Ввод информации осуществляется по команде IN port без запоминания информации;

- режим 1 - режим асинхронного обмена информацией или обмена по готовности данных между МП и ВУ по двум каналам A и B;

- режим 2 - режим стробируемого двунаправленного обмена, в котором можно осуществлять двунаправленную передачу информации только по одному каналу A.

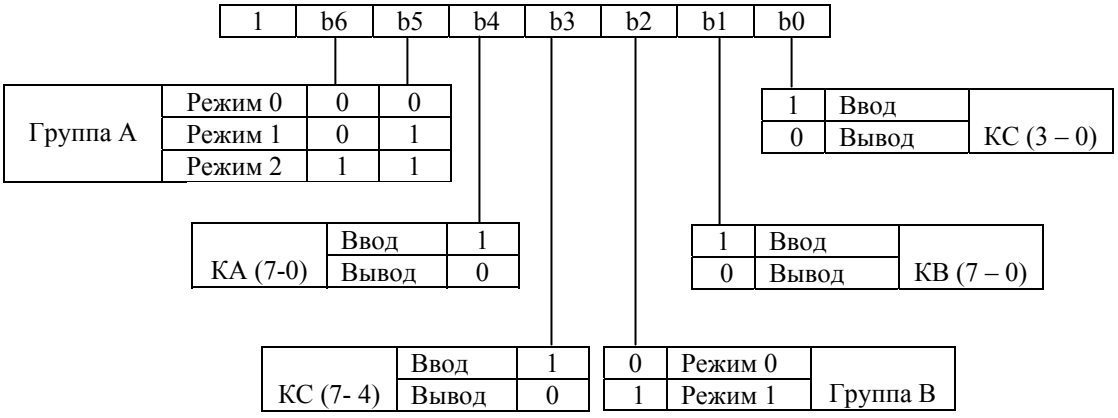


Рис. 4.14. Формат управляющего слова БИС КР580ВВ55

Программирование ППИ КР580ВВ55 осуществляется командой OUT port. В ее адресной части указывается двоичный код адреса регистра управления xxxxxxx11. Если в исследуемой системе используется несколько БИС

параллельного интерфейса, то 6 старших разрядов кода порта являются кодом номера соответствующей БИС КР580ВВ55.

Обмен данными при использовании ППИ выполняется командами ввода IN port и вывода OUT port, в которых значение port является адресом соответствующего регистра ППИ.

Изучение программируемого параллельного интерфейса КР580ВВ55 выполняется на стенде с помощью блоков статической и динамической индикации, реализованных на основе 4 семисегментных индикаторов LED MONITOR и СБИС КР580ВВ55.

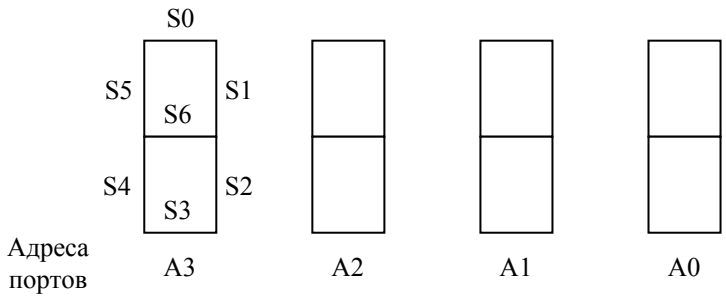


Рис.4.13. Кодирование отдельных сегментов семисегментных индикаторов и адреса портов четырехразрядного дисплея

Семисегментные индикаторы позволяют отображать все десятичные цифры и ряд букв русского и латинского алфавитов. Для поджига i-того сегмента s_i (рис.4.15) необходимо на соответствующий вход индикатора подать сигнал лог.1. Кодирование отображаемого символа осуществляется восьмиразрядным двоичным кодом b7b6b5b4b3b2b1b0, в котором значения бит b_i соответствуют сигналам управления сегментами s_i светодиодного индикатора. Например, цифре 0 соответствует код 00111111. Запись управляющего кода в соответствующий порт индикатора осуществляется при выполнении команды OUT port. В частности, для индикации 0 на индикаторе A0 необходимо записать семисегментный код цифры 0 в аккумулятор, например командой MVI A, 00111111, а затем командой OUT 373 загрузить содержимое аккумулятора в регистр этого индикатора.

Упрощенные структурные схемы блоков статической и динамической индикации представлены на рис.4.16 и 4.17.

Блок статической индикации

В состав блока статической индикации входят две БИС КР580ВВ55.. Каждый из шести семисегментных индикаторов подключается к системной шине с помощью отдельного канала (регистра) ППИ. Адреса каналов указаны на рис.4.16. Для отображения выводимой информации достаточно с помощью команды OUT port записать в соответствующий регистр требуемые данные. Адреса портов (регистров) ППИ указаны на рис. 4.16. Правильная работа блока индикации обеспечивается при предварительной инициализации ППИ, в которой программно задаются режимы работы каналов. Инициализация

производится один раз в начале программы вывода информации и в процессе дальнейшей работы блока индикации не повторяется. Настройка на режим синхронного вывода осуществляется следующей последовательностью команд:

```
MVI A,  200
OUT 003
OUT 007
```

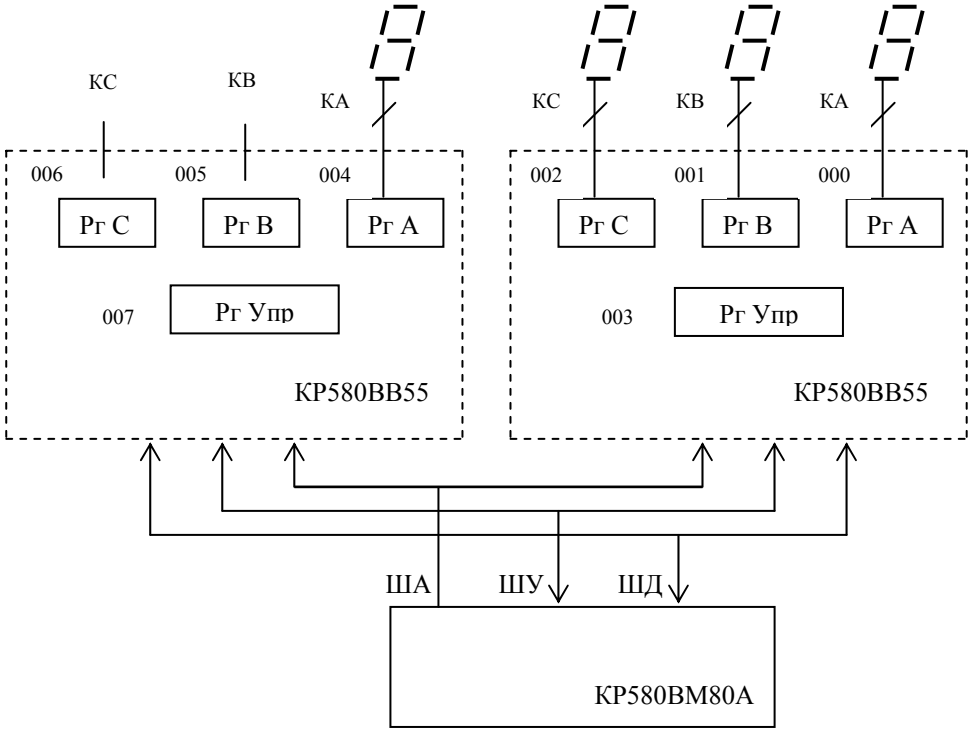


Рис.4.16. Структурная схема блока статической индикации

При использовании статической индикации аппаратные затраты пропорциональны объему отображаемой информации

Блок динамической индикации

Для заданного числа индикаторов аппаратную реализацию блока индикации можно упростить, если для отображения информации использовать принцип динамической индикации. Этот принцип основывается на инерционности человеческого зрения. Если отображаемую информацию предъявлять наблюдателю с определенной периодичностью (не менее 30 раз в сек), то у него создается впечатление постоянно включенного (немигающего) изображения.

Благодаря этому свойству индикаторы можно подключать к источнику информации не постоянно, а с определенной периодичностью на некоторое фиксированное время, при этом в промежутках между подключениями индикаторов по одним и тем же информационным линиям можно передавать коды управления другим индикаторам. Естественно, при реализации динамической индикации программная поддержка блока индикации усложняется.

В блоке динамической индикации (рис.4.17) четыре семисегментных индикатора подключены к выходам регистров каналов А и В БИС ППИ. Код индицируемого символа, формируемый на выходе канала А при выполнении команды OUT port, параллельно поступает на входы сегментов всех шести индикаторов. Адреса портов (регистров) ППИ указаны на рис. 4.17. Выбор адресуемого индикатора осуществляется с помощью дешифратора, включенного на выходе регистра канала В. Для адресации индикатора в порт В необходимо занести соответствующую информацию (реализуется командой OUT port). Как и при использовании статической индикации, ППИ блока динамической индикации предварительно должен быть проинициализирован.

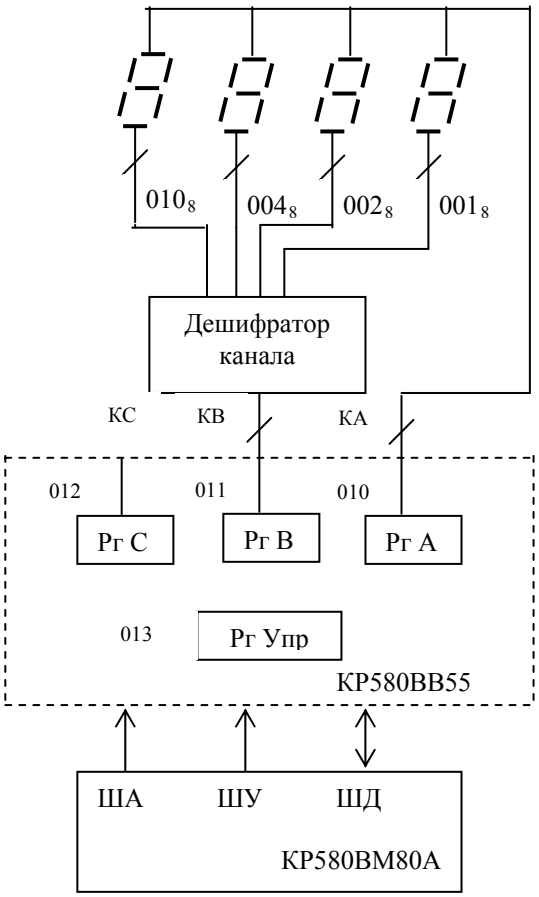


Рис.4.17. Структурная схема блока динамической индикации

Результаты исследования однокристального МП с фиксированным списком команд представьте в виде отчета. Содержание отчета определено в приложении 7.

4.2. Секционные (модульные) микропроцессоры с микропрограммным управлением серии К1804 с естественной, принудительной и стековой адресацией микрокоманд

Целью работ данного цикла является практическое ознакомление с основными модулями микропроцессорного комплекта серии К1804, организацией и методикой построения микропроцессорных устройств на его основе, приобретение навыков программирования микропроцессоров данного класса. Достижение указанной цели реализуется с помощью лабораторного стенда DiLaB 1804.

4.2.1.Структура и режимы работы лабораторного стенда DiLaB 1804

Лабораторный стенд DiLaB 1804 реализован на базе универсального стенда DiLaB со встроенной моделью ядра микропроцессорной системы с секционным микропроцессором. Внутренняя организация стенда соответствует типовой структуре МП-системы, выполненной на СБИС микропроцессорного комплекта К1804. Моделируемое ядро МП-системы объединяет модель блока центрального процессора, состоящего из одной 4-разрядной секции ЦПЭ К1804ВС1, регистра состояния с флагами результатов, мультиплексора сдвига и регистра выходных данных К1804ИР1, и модель блока микропрограммного управления в составе двух 4-разрядных секций схемы управления адресом микрокоманд (СУАМ) К1804ВУ1, ЗУ микрокоманд емкостью 256 36-разрядных слова, 36-разрядного регистра микрокоманд и схемы управляющих сигналов СУАМ, реализованной в виде ПЗУ емкостью 32 восьмиразрядных слова. Для обеспечения возможности оперативного программирования память микрокоманд выполнена в виде ОЗУ. Эмуляция модели ядра микропроцессорной системы реализована с помощью СБИС программируемой логики (ПЛИС EP2C8F256-8). Функциональная схема стенда DiLaB 1804 (рис. 4.18) кроме перечисленных блоков содержит устройство ввода информации, реализованное в виде клавишных переключателей адреса, данных и других элементов управления, обеспечивающих загрузку информации в ЗУМК и синхронизацию работы различных устройств тренажера, и устройство вывода информации, представленное в виде набора светодиодных индикаторов и табло жидкокристаллических индикаторов (ЖКИ), отражающих состояние выходных шин различных элементов процессорного блока, содержимое ячеек памяти микрокоманд и регистра микрокоманд.

Для управления работой стенда в различных режимах на его панели управления размещены следующие элементы управления и индикации:

- 4 кнопки-клавиши PUSH BUTTENS (UP, DOWN, LEFT, RIGHT);
- 8 переключателей SWITCHES (SW1 – SW8);
- 16-кнопочная клавиатура (KEYPAD);
- 8 светодиодных индикаторов LED INDICATORS (LED1 – LED7);
- 4 семисегментных индикатора LED MONITOR (Digit1 – Digit4);
- 2-строчный ЖКИ-монитор LCD MONITOR.

Назначение элементов управления и индикации поясняется ниже при описании режимов работы стенда DiLaB 1804.

Основными режимами работы стенда DiLaB 1804 являются:

- режим ВЫПОЛНЕНИЕ ПРОГРАММЫ (режим РАБОТА)
- режим ЗАГРУЗКА.

В режиме ЗАГРУЗКА выполняется ввод микрокоманд в память микрокоманд (микропрограммную память). В режиме РАБОТА осуществляется выполнение микропрограммы, хранящейся в микропрограммной памяти.

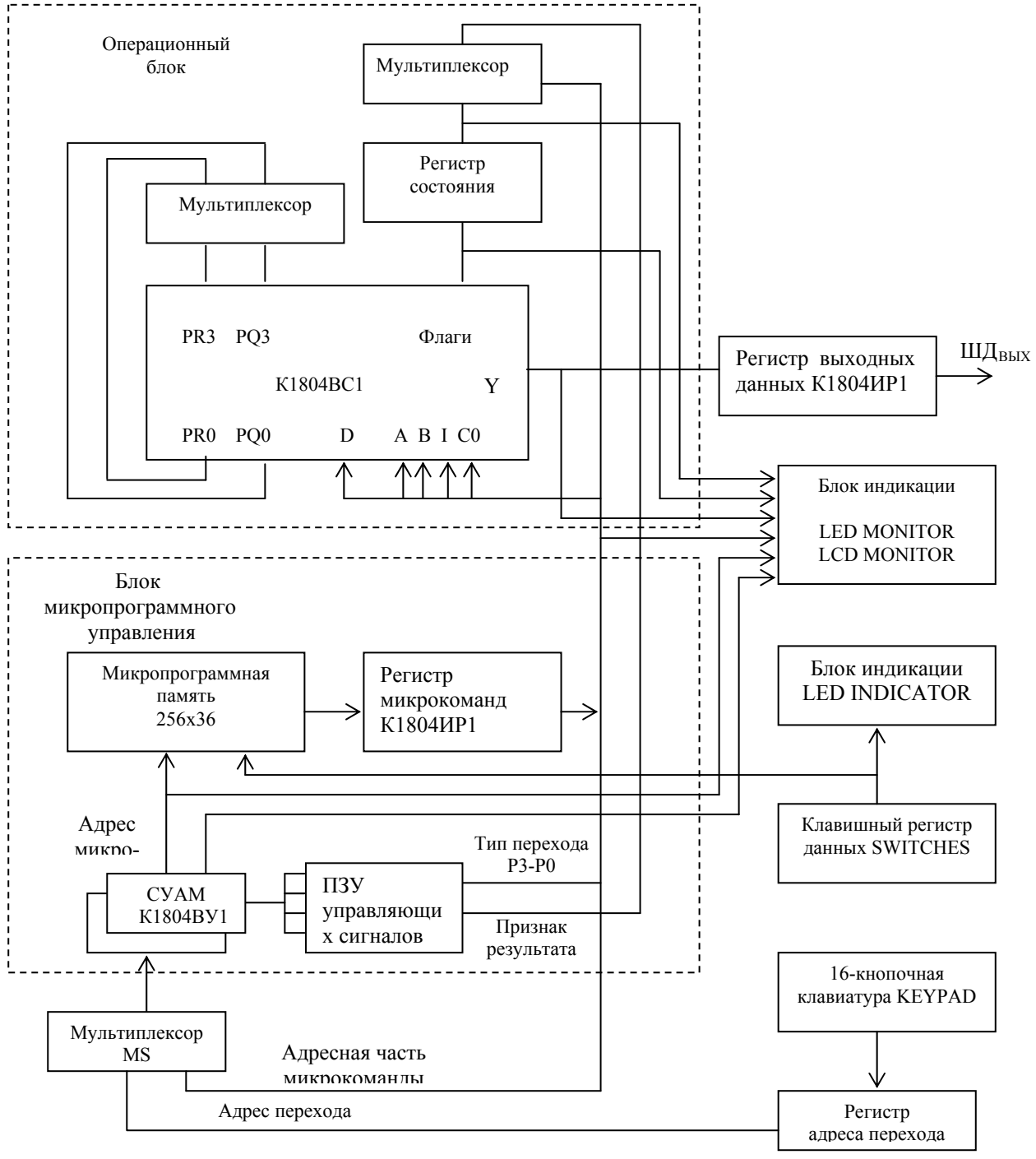


Рис. 4.18. Упрощенная функциональная схема стенда DiLaB 1804

Режим **ЗАГРУЗКА** задается при нажатии **кнопки UP,** а режим **РАБОТА** - при нажатии **кнопки DOWN (РАБОТА)**. Режим работы стенда отображается на Digit4 LED MONITOR. Символом **Р** отображается режим РАБОТА, а символом **З** - режим ЗАГРУЗКА.

В режиме РАБОТА центральный процессор в составе микротренажера обеспечивает выполнение микрокоманд, предварительно записанных в микропрограммную память. Исполняемые микрокоманды имеют 36-битный формат, функционально разбиваемый на 9 тетрад (рис. 4.19).

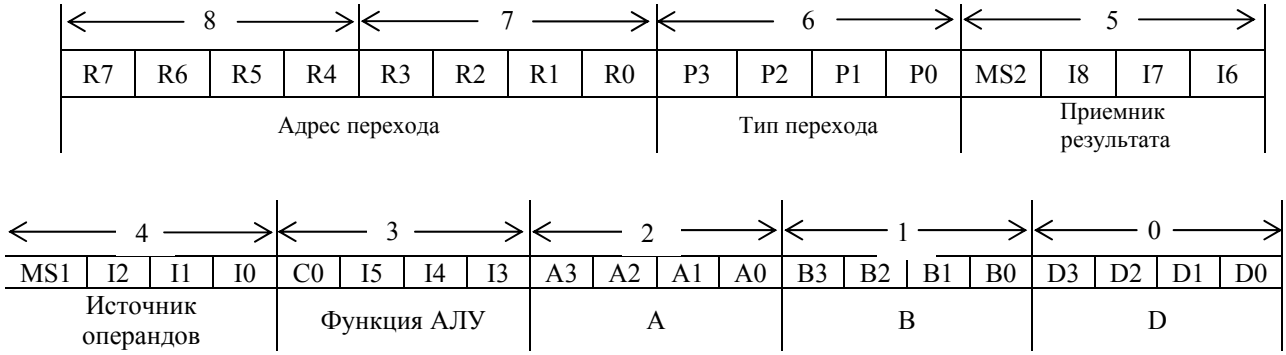


Рис.4.19. Формат микрокоманды, используемый в стенде DiLaB 1804

Формат исполняемых микрокоманд содержит коды управления ЦПЭ К1804ВС1 (операционное поле) и коды управления блоком БМУ, определяющим адрес следующей микрокоманды (адресное поле). Содержимое операционного и адресного полей отображается на экране ЖКИ (рис. 4.20).

Позиция символа в 1-й строке дисплея	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Мнемоника отображаемой информации	C	Д	E		P3	P2	P1	P0	MS2	I8	I7	I6	MS1	I2	I1	I0
Отображаемая информация	Адрес перехода (десятичный код)				Двоичный код типа перехода				Двоичный код приемника результата				Двоичный код источника операндов			
Позиция символа во 2-й строке дисплея	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
Мнемоника отображаемой информации	C0	I5	I4	I3	A3	A2	A1	A0	B3	B2	B1	B0	D3	D2	D1	D0
Отображаемая информация	Функция АЛУ (двоичный код)				Двоичный код поля адреса А				Двоичный код поля адреса В				Входные данные (двоичный код)			

Рис.4.20. Отображаемая информация о записываемой команде в режиме ЗАГРУЗКА

Запись микрокоманды в микропрограммную память осуществляется потетрадно в режиме ЗАГРУЗКА. Для записи микрокоманды необходимо задать адрес редактируемой микрокоманды в микропрограммной памяти и выполнить определенную последовательность действий:

- адрес редактируемой микрокоманды задается с помощью 16-кнопочной клавиатуры KEYPAD. Для задания адрес микрокоманды необходимо последовательно нажать кнопку «*» и три цифры-клавиши номера в десятичном формате (от старшего разряда десятичного кода номера к младшему). Десятичный код адреса микрокоманды отображается на 3-х младших индикаторах LED MONITOR Digit3 – Digit1. Адрес микрокоманды запоминается в специальном регистре адреса перехода $Р_{адр}$. В режиме РАБОТА на индикаторе адреса микрокоманды высвечивается адрес, который был помещен в $Р_{адр}$ последним. В режиме РАБОТА содержимое $Р_{адр}$ используется как адрес перехода при выполнении микрокоманд перехода по содержимому регистра $Р_{адр}$.

- потетрадная запись адресуемой микрокоманды осуществляется с использованием переключателей SW1 – SW4, SW5 – SW8 и кнопки RIGHT. Номер тетрады микрокоманды, содержимое которой контролируется или модифицируется, задается переключателями SW1 – SW4. Двоичный код номера тетрады индицируется на светодиодных индикаторах LED1 – LED4. Записываемые в выбранную тетраду данные задаются с помощью переключателей SW5 – SW8. Код записываемых данных отображается на светодиодных индикаторах LED5 – LED8 (горящий светодиод соответствует сигналу логической «1», погашенный светодиод - сигналу логического «0»). Адрес перехода, записываемый в две старшие тетрады микрокоманды, программно преобразуется в десятичный формат и в виде 3-разрядного десятичного кода отображается на дисплее ЖКИ (см. рис. 4.20).

- непосредственная запись данных в выбранную тетраду реализуется при нажатии кнопки RIGHT, которую можно назвать кнопкой ЗАПИСЬ.

Выборка микрокоманд из микропрограммной памяти и их выполнение ЦПЭ осуществляется под управлением блока микропрограммного управления в режиме РАБОТА. В этом режиме при нажатии кнопки LEFT формируются тактовые импульсы, синхронно с которыми очередная микрокоманда из микропрограммной памяти переписывается в регистр микрокоманд и выполняется. Микротренажер поддерживает два режима выполнения микрокоманд: пошаговое исполнение, реализуемое при кратковременном (однократном) нажатии кнопки LEFT, и непрерывное (автоматическое) выполнение микрокоманд, реализуемое при нажатии и удерживании кнопки LEFT. В соответствии с выполняемой функцией кнопку LEFT можно назвать кнопкой пошагового исполнения ПУСК. Завершение программы происходит при исполнении микрокоманды ОСТАНОВА (см. п. 4.2.2, микропрограмма 1).

Важнейшим информационным индикатором стенда является ЖКИ-индикатор (LCD MONITOR). С его помощью в режиме ЗАГРУЗКА контролируется результат операции загрузки микрокоманды в

микропрограммную память и отображается результат выполнения очередной (текущей) микрокоманды при пошаговом исполнении микропрограммы в режиме РАБОТА.

В режиме ЗАГРУЗКА на двухстрочный экран ЖКИ выводится содержимое 36-разрядной микрокоманды, записываемой в микропрограммную память: адрес перехода R7 - R0, представленный в десятичном формате (сотни, десятки, единицы), и двоичные коды первых 7 тетрад микрокоманды (рис. 4.20).

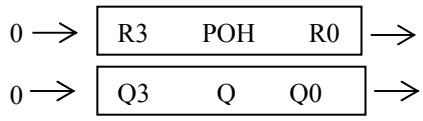
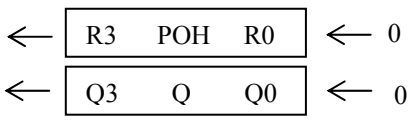
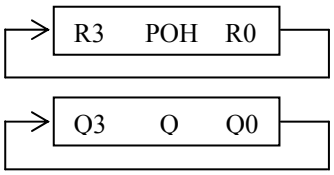
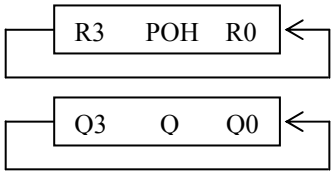
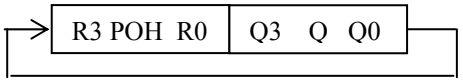
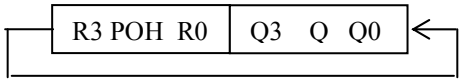
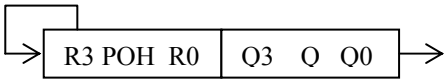
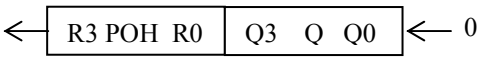
Операционное поле микрокоманды содержит 6 тетрад (тетрады 0 - 5). Нулевая тетрада используется для задания входных данных D. Входные данные D3-D0 отображаются во 2-й строке дисплея ЖКИ символами с номерами 29-32. В первой и второй тетрадах микрокоманды указываются адреса регистров регистрового ЗУ, задаваемые на одноименных входах B и A ЦПЭ соответственно. Двоичные коды адресов A и B отображаются во 2-й строке дисплея ЖКИ символами с номерами 21-24 и 25-28. Три младших разряда 3-й тетрады микрокоманды определяют код управления функцией АЛУ. Эти разряды управляют входами I5-I3 ЦПЭ (табл.3.1). Старший разряд 3-й тетрады задает входной перенос C0, управляя одноименным входом ЦПЭ. Содержимое 3-й тетрады микрокоманды отображается в 1-й строке дисплея ЖКИ символами с номерами 17-20. Три младших разряда 4-й тетрады микрокоманды кодируют выбор источников операндов. Выходы этих разрядов управляют входами I2-I0 ЦПЭ (табл.3.2). Управляющий код ЦПЭ I2-I0 отображается в 1-й строке дисплея ЖКИ символами с номерами 13-15. Три младших разряда 5-й тетрады содержат код управления приемником результата (табл.3.3). Кроме того, в этих же разрядах закодировано направление сдвига в сдвигателях на входе регистрового ЗУ и регистра Q. Выходы этих разрядов управляют входами I8-I6 ЦПЭ. Управляющий код ЦПЭ I8-I6 отображается в 1-й строке дисплея ЖКИ символами с номерами 10-12. В микротренажере в зависимости от кода микрокоманды могут быть реализованы сдвиговые операции с данными как 4-разрядной, так и 8-разрядной длины. Тип сдвиговой операции задается старшими разрядами 4-й и 5-й тетрад микрокоманды (разрядами 19 и 23), обозначенными как MS1 и MS2 (табл. 4.3). Сигналы MS2 и MS1 управляют внешним мультиплексором сдвига. На табло ЖКИ сигналы MS2 и MS1 отображается в 1-й строке дисплея ЖКИ символами с номерами 9 и 13.

Адресная часть микрокоманды, размещенная в ее трех последних тетрадах, определяет способ формирования адреса следующей микрокоманды при условных и безусловных переходах – код инструкции (тип) перехода (6-я тетрада) и собственно адрес перехода (7-я и 8-я тетрады). Адрес перехода (содержимое 7-ой и 8-ой тетрад) отображается 3-разрядным десятичным кодом в диапазоне от 0 до 255.

В зависимости от комбинации входных управляющих сигналов блок управления адресом микрокоманды К1804ВУ1 обеспечивает реализацию двух основных режимов адресации ЗУМК: принудительного, когда адрес следующей ячейки задается в текущей микрокоманде, и естественного, при котором адрес следующей микрокоманды получается сложением единицы с текущим адресом.

Код инструкции (тип) перехода P0-P3 (разряды 24-27 микрокоманды) и признак результата (C4,OVR,F3 или Z), формируемый на выходе мультиплексора регистра состояния (рис.4. 18), являются входными (адресными) сигналами ПЗУ емкостью 32 восьмиразрядных слова. Каждое слово ПЗУ хранит набор управляющих сигналов для схемы управления адресом микрокоманд К1804ВУ1.

Таблица 4.3

Код		Сдвиг вправо	Сдвиг влево
MS2	MS1		
0	0	Сдвиг 4-разрядного слова с вводом лог.0 в старшие разряды РОН и регистр Q 	Сдвиг 4-разрядного слова с вводом лог.0 в младшие разряды РОН и регистр Q 
0	1	Циклический сдвиг 4-разрядного слова 	Циклический сдвиг 4-разрядного слова 
1	0	Циклический сдвиг 8-разрядного слова 	Циклический сдвиг 8-разрядного слова 
1	1	Арифметический сдвиг 8-разрядного слова влево с вводом знакового разряда в старший разряд РОН 	Арифметический сдвиг 8-разрядного слова с вводом лог.0 в младший разряд регистра Q 

На табло ЖКИ двоичный код типа перехода P3-P0 отображается в 1-й строке дисплея ЖКИ символами с номерами 5-8. Адрес перехода, содержащийся в 7-й и 8-й тетрадах микрокоманды, отображается в 1-й строке дисплея ЖКИ **3-разрядным десятичным кодом** в позиции 1-3. Типы возможных переходов в зависимости от кода инструкции микрокоманды (типа перехода P3-P0) представлены в табл. 4.4.

Таблица 4.4

Код инструкции перехода				Тип перехода
P3	P2	P1	P0	
0	0	0	0	Переход на адрес из регистра микрокоманд, если Z=0
0	0	0	1	Переход на адрес из регистра микрокоманд
0	0	1	0	Продолжить (переход на следующий адрес)
0	0	1	1	Переход на адрес, формируемый клавишным регистром адреса (переход на адрес вектора)
0	1	0	0	Переход к подпрограмме, если Z=0
0	1	0	1	Переход к подпрограмме
0	1	1	0	Возврат из подпрограммы
0	1	1	1	Переход по стеку
1	0	0	0	Окончить цикл и вытолкнуть из стека, если Z=1
1	0	0	1	Загрузить стек и продолжить
1	0	1	0	Вытолкнуть из стека и продолжить
1	0	1	1	Окончить цикл и вытолкнуть из стека, если C4=1
1	1	0	0	Переход на адрес из регистра микрокоманд, если Z=1
1	1	0	1	Переход на адрес из регистра микрокоманд, если F3=1
1	1	1	0	Переход на адрес из регистра микрокоманд, если OVR=1
1	1	1	1	Переход на адрес из регистра микрокоманд, если C4=1

В режиме РАБОТА на табло ЖКИ-монитора отображается информация о состоянии выходов различных блоков микротренажера при исполнении очередной микрокоманды (рис. 4.21).

Позиция символа в 1-й строке дисплея	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Мнемоника отображаемой информации	C	Д	E		C	Д	E		d3	d2	d1	d0		C4	OV	F3
Отображаемая информация	Десятичный адрес текущей МК				АДРЕС Десятичный адрес следующей МК				ДАННЫЕ Двоичный код результата на выходе ЦПЭ					Флаги результата на выходе ЦПЭ		
Позиция символа во 2-й строке дисплея	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
Мнемоника отображаемой информации	Z		PQ3	PQ0	PR3	PR0		d3	d2	d1	d0		C	OV	F3	Z
Отображаемая информация	Флаг нуля		Входы/Выходы сдвига ЦПЭ					Двоичный код результата на выходе регистра данных					Флаги результата предыдущей МК на выходе регистра состояния			

Рис. 4.21. Отображаемая информация о состоянии блоков микротренажера при исполнении микрокоманды в режиме РАБОТА

Проиллюстрируем работу стенда DiLaB 1804 на примере выполнения микропрограммы, размещенной в микропрограммной памяти, начиная с адреса 000.

Исполнению микропрограммы предшествует запись микропрограммы в память (этап загрузки). Последовательность действий при записи или модификации микрокоманд в памяти следующая:

- стенд переводится в режим ЗАГРУЗКА (реализуется однократным нажатием кнопки UP). Установленный режим работы контролируется по индикатору Digit4 LED- MONITORa (на семисегментном LED-индикаторе должен высветиться символ **3**;

- с помощью 16-кнопочной клавиатуры KEYPAD устанавливается адрес микрокоманды (записываемой или редактируемой). Для задания адреса микрокоманды на клавиатуре необходимо последовательно нажать кнопку (*) и три цифры-клавиши номера в десятичном формате (от старшего разряда десятичного кода номера к младшему, в данном случае 000). Адрес микрокоманды в десятичном формате высвечивается на 3-х младших индикаторах LED MONITORa Digit3 – Digit1 (при несоответствии задание адреса выполняется повторно);

- при нажатии клавиши LEFT (ПУСК) содержимое микрокоманды, размещенной в памяти по установленному адресу, переписывается в регистр микрокоманд и высвечивается на экране ЖКИ-монитора (в формате, показанном на рис. 4.20). При необходимости выполняется редактирование микрокоманды.

Содержимое микрокоманды модифицируется потетрадно. Потетрадная запись осуществляется с использованием переключателей SW1 – SW4, SW5 – SW8 и клавиши RIGHT (ЗАПИСЬ). Для модификации содержимого тетрады необходимо

- установить номер тетрады (двоичный номер тетрады задается переключателями SW1 – SW4 и индицируется на светодиодных индикаторах LED1 – LED4),

- набрать на переключателях SW5 – SW8 двоичный код записываемых данных (код записываемых данных отображается на светодиодных индикаторах LED5 – LED8);

- нажать клавишу RIGHT (ЗАПИСЬ). При нажатии клавиши RIGHT содержимое тетрады (набранные данные) переписывается в память. Значение данных в памяти контролируется в соответствующем поле экрана ЖКИ-монитора.

Внимание! Запись кода адреса микрокоманды (в 7-ю и 8-ю тетрады) осуществляется 8-разрядным двоичным кодом, а отображение адреса происходит в десятичном коде. Преобразование кода реализуется внутренними схемами микротренажера.

После окончания редактирования микрокоманды по искомому адресу аналогичным образом осуществляется редактирование других микрокоманд микропрограммы.

Выполнение микропрограммы реализуется в режиме РАБОТА. Перед выполнением необходимо задать стартовый адрес микропрограммы и загрузить содержимое 1-й микрокоманды в регистр микрокоманд. Указанные операции реализуются в режиме ЗАГРУЗКА описанным выше способом. После их выполнения микротренажер переводится в режим РАБОТА (кратковременным нажатием клавиши DOWN). Установленный режим работы контролируется по индикатору Digit4 LED- MONITORa, на котором должен высветиться символ **Р**. При переключении в режим РАБОТА происходит перекоммутирование сигналов на входе ЖКИ-индикатора. В этом режиме на выходе ЖКИ-индикатора отображается информация о состоянии выходов различных блоков микротренажера при исполнении очередной микрокоманды (рис. 4.21).

Собственно выполнение микрокоманды происходит при нажатии клавиши ПУСК. При последовательных кратковременных нажатиях клавиши ПУСК осуществляется последовательное выполнение микрокоманд, и пользователь по содержимому табло ЖКИ-индикатора может контролировать результаты выполнения отдельных микрокоманд. При нажатии и удерживании клавиши ПУСК происходит автоматическое выполнение микропрограммы до команды останова. Результат выполнения микропрограммы отображается на табло ЖКИ-индикатора.

Обратите ***внимание***, что при первом шаге выполнения микропрограммы (при первом нажатии клавиши ПУСК (LEFT)) на табло ЖКИ не отображается содержимое младших тетрад состояния блоков лабораторного стенда. При последующих нажатиях клавиши ПУСК происходит полное отображение состояния работы блоков стенда. С чем это связано?

4.2.2. Программа изучения СБИС МПК К1804 на стенде DiLaB 1804

Работа рассчитана на несколько посещений и предполагает:

- изучение списка микрокоманд ЦПЭ К1804ВС1 и методики его программирования;
- рассмотрение принципов построения БМУ на основе БИС К1804ВУ1 и К1804ВУ3;
- приобретение навыков программирования и построения простейших контроллеров на МП серии К1804.

Оценить особенность архитектуры МПК К1804, его функциональные возможности, приобрести навыки программирования устройств на основе МПК К1804 позволяют восемь микропрограмм, тексты которых приводятся ниже.

Микропрограмма 1 иллюстрирует методы загрузки данных в РОН ЦПЭ. Текст программы приведен в табл. 4.5.

Таблица 4.5

Адреса памяти	Микрокоманда								Выполняемая функция
	Адрес перехода	Тип перехода	Приемник результата	Источник операнда	Функция АЛУ	A	B	D	
	7-8	6	5 MS2 I8-I6	4 MS1 I2-I0	3 C0 I5-I3	2	1	0	
000	xxxx	0010	x 011 (F → B)	x 111 (R=D S=0)	x 011 (RvS)	xxxx	0000	0010	Загрузка R0
001	xxxx	0010	x 001 (Y=F)	x 011 (R=0 S=B)	x 011 (RvS)	xxxx	0000	xxxx	Чтение R0
002	xxxx	0010	x 011	x 111	x 011	xxxx	0001	0100	Загрузка R1
003	xxxx	0010	x 001	x 011	x 011	xxxx	0001	xxxx	Чтение R1
004	xxxx	0010	x 011	x 111	x 011	xxxx	1001	0101	Загрузка R9
005	005	0001	x 001	x 011	x 011	xxxx	1001	xxxx	Чтение R9

После занесения микропрограммы в память необходимо:

- выполнить микропрограмму в пошаговом режиме, фиксируя результат выполнения отдельных микрокоманд по содержимому поля ДАННЫЕ

(«Двоичный код результата на выходе ЦПЭ») ЖКИ-монитора (позиции символов 9 – 12);

- убедиться, что с помощью указанных микрокоманд можно загрузить любой из шестнадцати РОН произвольным числом, меняя только содержимое полей B и D.

Микрокоманда с адресом 005 в этой микропрограмме, передающая управление самой себе (в поле инструкции перехода такой микрокоманды должен быть указан код 0001, а в поле адреса перехода - адрес микрокоманды), является микрокомандой «останова». С помощью индикаторов ДАННЫЕ убедитесь, что микропрограмма «зацикливается» на микрокоманде с адресом 005, при этом любое последующее нажатие кнопки ПУСК не изменяет содержимое счетчика микрокоманд БИС К1804ВУ1.

Загрузка регистра Q выполняется аналогичными командами, однако для этого необходимо изменить приемник результата. Выполните эту микрооперацию.

Микропрограмма 2 иллюстрирует выполнение сдвиговых операций операндов 4-разрядной и 8-разрядной длины. Текст программы приведен в табл. 4.6.

Таблица 4.6

Адрес памяти	Микрокоманда								Выполняемая операция
	7-8	6	5 MS2 18-16	4 MS1 12-10	3	2	1	0	
000	xxxx	0010	x 011 F→B	x 111 R=D S=0	x 011 RvS	xxxx	0000	0010	Загрузка R0
001	xxxx	0011	0 111 2F→B	1 011 R=0 S=B	x 011 RvS	xxxx	0000	xxxx	Сдвиг R0 влево циклический.
002	xxxx	0011	0 101 F/2→B	1 011 R=0 S=B	x 011 RvS	xxxx	0000	xxxx	Сдвиг R0 вправо циклический
003	xxxx	0010	x 000 F→Q	X 010 R=0 S=Q	x 100 R ∧ S	xxxx	xxxx	xxxx	Обнуление Q
004	xxxx	0011	1 110 2F→B 2Q→Q	0 011 R=0 S=B	x 011 R ∨ S	xxxx	0000	xxxx	Циклический сдвиг двойной длины влево
005	xxxx	0011	1 100 F/2→B Q/2→Q	0 011 R=0 S=B	x 011 RvS	xxxx	0000	xxxx	Циклический сдвиг двойной длины вправо
006	006	0001	x 001 Y=F	x 011 R=0 S=B	x 011 RvS	xxxx	xxxx	xxxx	Пустая операция

В данную микропрограмму включены микрокоманды (1, 2, 3, 5 и 6), в которых реализуется переход по адресу, содержащемуся в регистре адреса перехода. Использование этих микрокоманд позволяет выполнить операцию, определяемую операционным полем микрокоманды, произвольное число раз. Для выполнения микропрограммы необходимо соответствующим образом управлять регистром адреса перехода (в этот регистр перед выполнением микропрограммы необходимо занести адрес микрокоманды, исполнение которой планируется многократно). Тип сдвиговых операций задается разрядами микрокоманды MS2 и MS1 (19 и 23).

Выполните микропрограмму 2, последовательно реализуя следующие действия:

- выполните микрокоманду, размещенную по нулевому адресу ЗУМК. В результате ее выполнения в РОН0 загружается число 2, а БМУ формирует адрес перехода к следующей микрокоманде, размещенной по первому адресу ЗУМК. На индикаторах ДАННЫЕ ЖКИ-индикатора высветится число 2. Поскольку микрокоманда, размещенная по первому адресу ЗУМК, является микрокомандой, осуществляющей переход по вектору, то при ее выполнении БМУ сформирует адрес, определяемый регистром адреса. Для правильного выполнения микропрограммы соответствующий адрес должен находиться в регистре адреса;
- перейдите в режим ЗАГРУЗКА и загрузите в регистр адреса перехода адрес микрокоманды, исполнение которой планируется многократно (в данном случае 001);
- вернитесь в режим РАБОТА и нажав клавишу ПУСК выполните микрокоманду, размещенную по первому адресу. Поскольку в регистре адреса хранится число 001, БМУ реализует переход на адрес 001, т.е. снова к микрокоманде, размещенной по первому адресу ЗУМК. При последовательном нажатии кнопки ПУСК микрокоманда, размещенная по первому адресу, будет выполнять циклический сдвиг содержимого РОН0 влево на один разряд. Зафиксируйте результат выполнения этой микрокоманды при нажатии клавиши ПУСК пять раз (значения индикаторов ДАННЫЕ ЖКИ-монитора). Сравните результат, отображаемый на табло, с теоретическим значением содержимого регистра при циклическом сдвиге. Объясните причину несоответствия.
- перейдите в режим ЗАГРУЗКА и загрузите в регистр адреса число 002 - адрес микрокоманды, реализующей циклический сдвиг вправо;
- вернитесь в режим РАБОТА и нажав клавишу ПУСК выполните микрокоманду, размещенную по адресу 2. Последовательно нажимая клавишу ПУСК и контролируя результат по индикаторам ДАННЫЕ, убедитесь в том, что при каждом нажатии ПУСК осуществляется циклический сдвиг содержимого РОН0 вправо на один разряд. Зафиксируйте результат выполнения этой микрокоманды при нажатии клавиши ПУСК несколько раз;
- аналогично рассмотренному загрузите в регистр адреса адрес 003 и выполните микрокоманду обнуления регистра Q, размещенную по этому адресу. При ее выполнении реализуется переход к следующей микрокоманде (по адресу 004), которая является микрокомандой перехода по вектору;
- аналогично рассмотренному загрузите в регистр адреса $P_{адр}$ адрес 004 и выполните микрокоманду циклического сдвига двойной длины влево. С помощью световых индикаторов ДАННЫЕ зафиксируйте результат выполнения этой микрокоманды при нажатии клавиши ПУСК несколько раз;
- аналогично рассмотренному загрузите в регистр адреса $P_{адр}$ адрес 005 и выполните микрокоманду циклического сдвига двойной длины вправо. С помощью индикаторов ДАННЫЕ ЖКИ-монитора зафиксируйте результат выполнения этой микрокоманды при нажатии клавиши ПУСК несколько раз;

- аналогично рассмотренному загрузите в регистр адреса $P_{адр}$ адрес 006 и завершите выполнение микропрограммы 2.

Объясните результаты выполнения микрокоманд сдвига.

Составьте и выполните микропрограмму, демонстрирующую выполнение арифметического сдвига 8-разрядной длины вправо и влево, а также 4-разрядного сдвига с вводом лог.0 в старшие (при сдвиге влево) или младшие (при сдвиге вправо) разряды РОН и регистра Q.

В МП К1804ВС1 при выполнении сдвиговых операций выдвигаемый бит не фиксируется. Определение значения выдвигаемого бита можно выполнить программным способом. Примеры определения значения выдвигаемого бита рассматриваются в микропрограмме 8.

Микропрограмма 3 иллюстрирует действие некоторых арифметических операций в ЦПЭ К1804ВС1. Текст микропрограммы приведен в табл. 4.7.

Микрокоманды 0 и 1 используются для инкремента и декремента содержимого РОН0. Здесь и при последующем изложении номер микрокоманды соответствует ее адресу. Микрокоманды 2, 3 демонстрируют сложение (вычитание) содержимого РОН с константой. Микрокоманды 4, 5, 6 реализуют сложение содержимого двух РОН. В данной микропрограмме, как и в предыдущей, используются микрокоманды, реализующие переход по вектору - микрокоманды 0, 1, 2, 3 и 4.

Таблица 4.7

Адреса памяти	Микрокоманда								Выполняемая функция
	7-8	6	5	4	3	2	1	0	
			MS2 I8-I6	MS1 I2-I0	C0 I5-I3				
000	xxxx	0011	x011	x011	1 000	xxxx	0000	xxxx	R0+1
001	xxxx	0011	x011	x011	0 001	xxxx	0000	xxxx	R0 - 1
002	xxxx	0011	x011	x101	0 000	0000	0000	0011	R0+3
003	xxxx	0011	x011	x101	1 001	0000	0000	0101	R0 – 5
004	xxxx	0010	x011	x011	x 100	xxxx	0000	0101	Обнуление R0
005	xxxx	0010	x011	x111	x 011	xxxx	0001	0101	R1=5
006	xxxx	0011	x011	x001	0 000	0001	0000	xxxx	R0← R0 + R1
007	xxxx	0011	x011	x001	0 000	0001	0000	xxxx	R0← R0 + R1
008	xxxx	0011	x011	X101	0 000	0000	xxxx	0111	R0← R0 + D
009	009	0001	x001	x011	x 011	xxxx	xxxx	xxxx	Пустая операция

Выполните микропрограмму 3. Результат выполнения отдельных микрокоманд фиксируйте с помощью индикаторов ДАННЫЕ ЖКИ-монитора.

Составьте и выполните микропрограмму, демонстрирующую выполнение других микрокоманд арифметических и логических операций.

Микропрограмма 4 иллюстрирует работу БМУ при безусловных переходах. Текст микропрограммы приведен в табл. 4.8. Микрокоманды 0, 4, 10, 15 выполняют безусловный переход по адресу из регистра микрокоманд, микрокоманды 3, 9 реализуют безусловный переход "Продолжить".

Поскольку при передачах управления операция, выполняемая ЦПЭ, обычно не имеет значения, кодирование разрядов операционного поля микрокоманд может быть любым. Однако для сохранения неизменным содержимого блока внутренних регистров в поле приемника результата микрокоманды необходимо указать код 001, запрещающий куда-либо запись результата АЛУ.

Таблица 4.8

Адреса памяти	Микрокоманда								Выполняемая функция
	7-8	6	5	4	3	2	1	0	
000	009	0001	x001	xxxx	xxxx	xxxx	xxxx	xxxx	Безусловный переход (БП) по адресу 9
003	xxxx	0010	x001	xxxx	xxxx	xxxx	xxxx	xxxx	БП (продолжить)
004	015	0001	x001	xxxx	xxxx	xxxx	xxxx	xxxx	БП по адресу 15
009	xxxx	0010	x001	xxxx	xxxx	xxxx	xxxx	xxxx	БП (продолжить)
010	003	0001	x001	xxxx	xxxx	xxxx	xxxx	xxxx	БП по адресу 3
015	000	0001	x001	xxxx	xxxx	xxxx	xxxx	xxxx	БП по адресу 0

Выполните микропрограмму 4 в пошаговом режиме. Результат выполнения микропрограммы 4, формируемый на выходе БИС К1804ВУ1, наблюдайте по содержимому поля АДРЕС («десятичный код адреса следующей микрокоманды») ЖКИ-монитора (позиции символов 5-7).

Результат выполнения микропрограммы представьте в виде диаграммы переходов.

Микропрограмма 5 иллюстрирует один из способов организации цикла с помощью стека. Текст микропрограммы приведен в табл. 4.9.

Таблица 4.9

Адреса памяти	Микрокоманда								Выполняемая функция
	7-8	6	5	4	3	2	1	0	
000	xxxx	0010	x001	xxxx	xxxx	xxxx	xxxx	xxxx	Продолжить
001	xxxx	1001	x001	xxxx	xxxx	xxxx	xxxx	xxxx	Загрузить в стек и продолжить
002	xxxx	0010	x001	xxxx	xxxx	xxxx	xxxx	xxxx	Продолжить
003	010	0001	x001	xxxx	xxxx	xxxx	xxxx	xxxx	БП по адресу 10
010	xxxx	0111	x001	xxxx	xxxx	xxxx	xxxx	xxxx	Переход по стеку

В данной микропрограмме, как и в предыдущей, действие, выполняемое ЦПЭ, не имеет значения, поэтому кодирование разрядов операционного поля микрокоманд может быть любым.

Выполните микропрограмму в пошаговом режиме, фиксируя адреса, формируемые на выходе БИС К1804ВУ1, по содержимому поля АДРЕС ЖКИ-монитора.

Отметим, что микропрограмма 5 не демонстрирует способы выхода из цикла. Для выхода из цикла можно использовать микрокоманды условных переходов или микрокоманды, в адресном поле которых (тетрада 6) закодирована инструкция перехода по результатам проверки конца цикла: -1000, если результат операции АЛУ равен нулю (Z=1) или 1011, если С4=1. В корректной микропрограмме одна из указанных микрокоманд обязательно должна быть включена в тело цикла.

Разработайте ***микропрограмму 6***, реализующую обращение к подпрограмме и возврат из нее, в соответствии с диаграммой переходов на рис. 4.22.

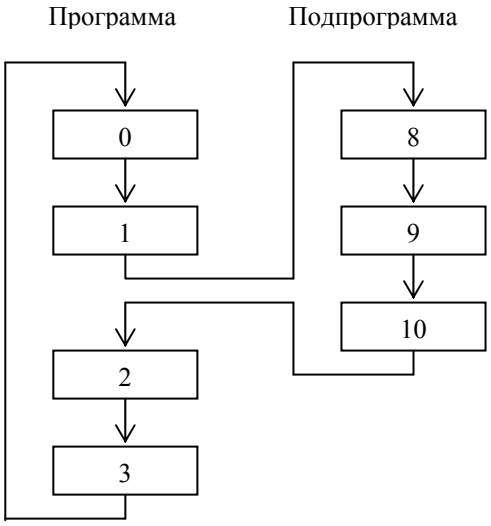


Рис.4.22. Диаграмма переходов
при вызове подпрограммы

В разрабатываемой микропрограмме используйте микрокоманду, в которой закодированы инструкция "Переход к подпрограмме" (код 0101 в поле 6 микрокоманды) и адрес первой микрокоманды подпрограммы (поле 7-8). В последней микрокоманде подпрограммы должна быть указана инструкция "Возврат из подпрограммы" (код 0110 в поле 6).

Выполните микропрограмму в пошаговом режиме, контролируя адреса, формируемые на выходе БИС К1804ВУ1 по содержимому поля АДРЕС ЖКИ-монитора.

Микропрограмма 7 иллюстрирует работу стека при вложении подпрограмм, когда одна подпрограмма вызывает другую, которая, в свою очередь, может вызвать третью и т.д. На рис.4.15 предложена диаграмма организации

последовательности выполнения вложенных микропрограмм. В табл. 4.10 содержится начальный фрагмент микропрограммы, соответствующий, диаграмме. ***Закончите написание текста микрограммы и выполните*** её в пошаговом режиме, контролируя адреса, формируемые на выходе БИС К1804ВУ1 по содержимому поля АДРЕС ЖКИ-монитора.

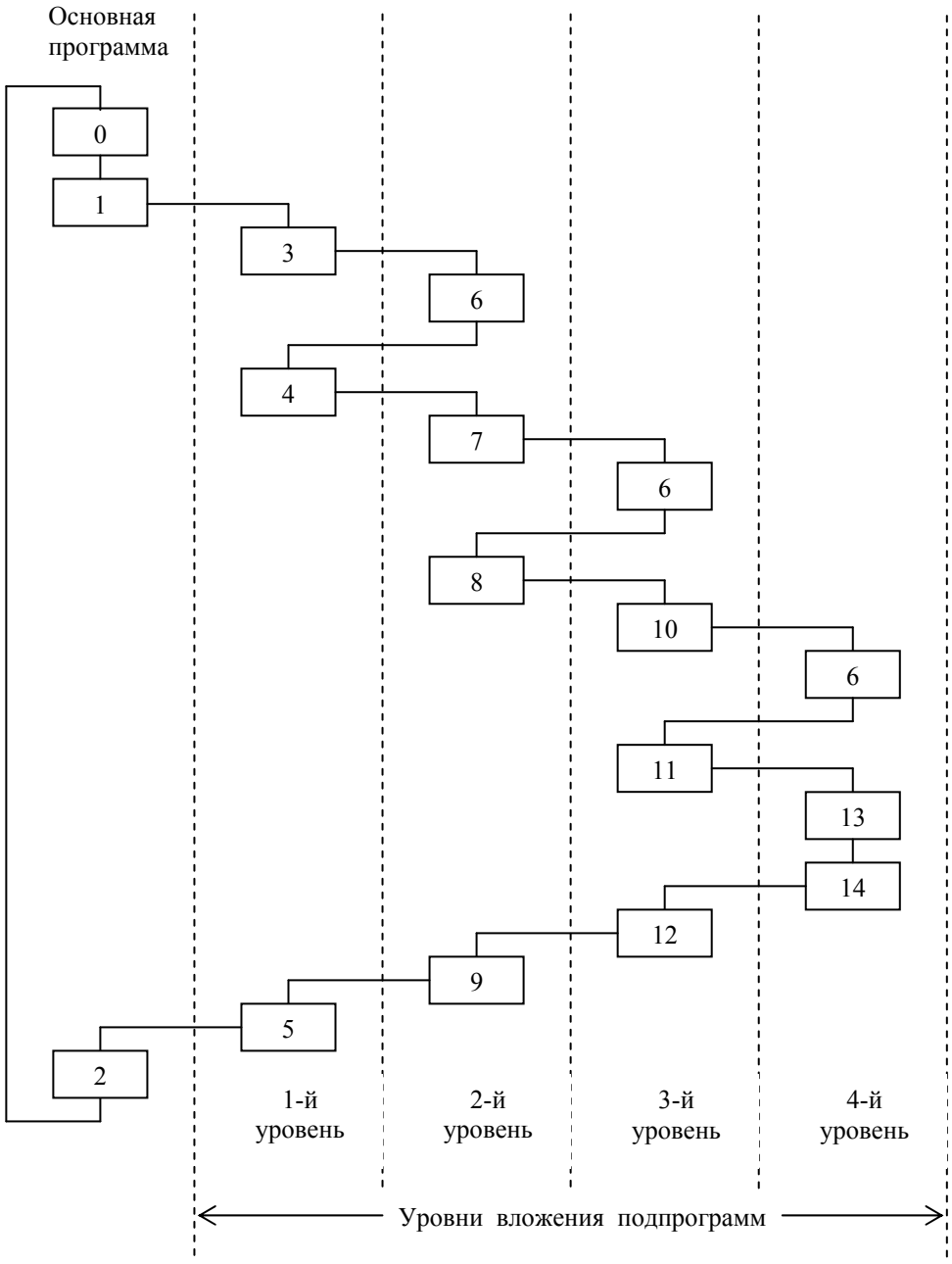


Рис. 4.15. Иллюстрация работы стека при вложении подпрограмм

Таблица 4.10

Адреса памяти	Микрокоманда								Выполняемая функция
	7-8	6	5	4	3	2	1	0	
000	xxx	0010	x001	xxxx	xxxx	xxxx	xxxx	xxxx	БП (продолжить)
001	003	0101	x001	xxxx	xxxx	xxxx	xxxx	xxxx	Переход к подпрограмме 1
002	000	0001	x001	xxxx	xxxx	xxxx	xxxx	xxxx	БП по адресу из Рг микрокоманд
003	006	0101	x001	xxxx	xxxx	xxxx	xxxx	xxxx	Переход к подпрограмме 2
004	007	0101	x001	xxxx	xxxx	xxxx	xxxx	xxxx	Переход к подпрограмме 3
005	xxx	0110	x001	xxxx	xxxx	xxxx	xxxx	xxxx	Возврат из подпрограммы 1

Выполните микропрограмму 7 в пошаговом режиме, контролируя адреса, формируемые на выходе БИС К1804ВУ1 по содержимому поля АДРЕС ЖКИ-монитора.

Ответьте на вопросы:

Какие ограничения на количество вложенных подпрограмм накладывает организация стека БИС СУАМ К1804 ВУ1?

Можно ли с использованием СУАМ К1804 ВУ1 реализовать микропрограмму, содержащую 10 подпрограмм?

Микропрограмма 8 иллюстрирует реализацию условных переходов в типовых микропрограммах. Текст микропрограммы представлен в табл. 4.11.

Таблица 4.11

Адреса памяти	Микрокоманда								Выполняемая функция
	7-8	6	5	4	3	2	1	0	
000	xxxx	0010	x011	x011	1000	xxxx	0000	xxxx	R0 +1, продолжить
001	1111	1111	x001	xxxx	xxxx	xxxx	xxxx	xxxx	УП по С4
002	0000	0001	x001	xxxx	xxxx	xxxx	xxxx	xxxx	БП по адресу 0
015	xxxx	0010	x001	xxxx	xxxx	xxxx	xxxx	xxxx	БП (продолжить)

В стенде DiLaB 1804 признаки результата выполненной в АЛУ операции запоминаются в регистре состояния. В соответствии со значениями этих признаков предусмотрено несколько различных команд условного перехода, кодирование которых в микрокомандах осуществляется согласно табл.4.4. Если условие выполнено, то реализуется переход по адресу, указанному в тетрадах 7-8 микрокоманды, в противном случае выполняется следующая по порядку микрокоманда. Конструктивной особенностью данного тренажера является то, что формируемые в АЛУ признаки результата записываются в регистр состояния в момент завершения импульса синхронизации. По этой причине реализацию условных переходов можно осуществить не ранее, чем в

следующем такте после микрокоманды, в которой формируется контролируемый признак.

Другой особенностью тренажера является то, что при выполнении микрокоманды, содержащей код условного перехода, признаки результата операции, указанной в этой микрокоманде, никуда не записываются, а потому и переход по ним невозможен. Наличие блокировки записи в регистр состояния при выполнении микрокоманд условных переходов позволяет осуществить несколько проверок результата выполненной операции по значению различных флагов, при этом содержимое регистра состояния будет оставаться неизменным. Содержимое регистра состояния отображается в одноименном поле ЖКИ-монитора (позиции символов 29-32).

Микрокоманда 0 выполняет инкремент содержимого регистра R0 и передает управление следующей микрокоманде. Микрокоманда 1 проверяет состояние выходного переноса АЛУ С4. При единичном значении флага С4 выполняется переход на адрес 15, в противном случае - к следующей микрокоманде. Операции, выполняемые ЦПЭ в микрокомандах 2 и 15, не имеют значения, поэтому кодирование разрядов операционного поля этих микрокоманд за исключением поля приемника результата может быть любым.

Выполните микропрограмму в пошаговом режиме, фиксируя адреса, формируемые на выходе БИС К1804ВУ1 по содержимому поля АДРЕС ЖКИ-монитора.

Для определения значения выдвигаемого бита можно выполнить следующие действия:

- при сдвиге влево для определения значения выдвигаемого бита до сдвига анализируется значение старшего разряда (например, микрокомандой чтения регистра), после этого выполняется сдвиг и проверка флага S. Если S=1, то выдвигаемый бит (перенос) равен 1, в противном случае – 0.
- при сдвиге вправо исходное слово перед сдвигом маскируется словом-маской 00…01, содержащем единственную 1 в младшем разряде слова. Затем выполняется сдвиг и реализуется условный переход по значению флага Z. Если Z=1, то выдвигаемый бит (перенос) равен 0 и осуществляется выбор одной из возможных ветвей продолжения, в противном случае – перенос равен 1 и осуществляется выбор другой ветви.

Составьте и выполните фрагменты микропрограмм, иллюстрирующие определение значения выдвигаемого бита при сдвигах вправо и влево.

4.2.3 Изучение принципов построения простейших МП-систем на СБИС серии К1804. Основы микропрограммирования таких систем

По указанию преподавателя составьте и выполните несколько микропрограмм индивидуального задания из списка заданий приложения 3. Учитывая ограниченный объем оперативной памяти стенда, в лаборатории возможно решение лишь простейших задач. Запишите эти микропрограммы в память, отладьте и выполните их в пошаговом режиме, контролируя

результаты их выполнения с помощью блока индикации. Протокол и результаты выполнения микропрограмм представьте преподавателю.

4.2.4. Разработка функциональной схемы микропроцессорной системы на СБИС К1804

Разработайте и представьте в отчете функциональную (структурную) схему микропроцессорной системы с 4n-разрядным ЦПЭ К1804ВС1 и БМУ, объединяющем m секций СУАМ К1804ВУ1. Значения n и m уточните у преподавателя. Приведите формат микрокоманды разрабатываемой системы.

Результаты исследования секционных микропроцессоров с микропрограммным управлением представьте в виде отчета. Содержание отчета определено в приложении 7

Список литературы

1. В. Ф. Мелехин. Вычислительные машины : учебник для студ. учреждений высш. проф. образования / В. Ф. Мелехин, Е. Г. Павловский. — М.: Издательский центр «Академия», 2013. — 384 с.

2. Е.Г.Павловский, В.А.Жвариков. Основы организации ЭВМ и микропроцессоров. Учебное пособие и методические указания к лабораторному практикуму. – СПб.: Санкт –Петербургский государственный политехнический университет, 2008. – 125 с.

Приложение 1

Система команд микропроцессоров КР580 и К1821

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 8080 ячейка памяти M трактуется как своеобразный регистр, обращение к которому производится только с использование 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. П1.

Таблица П1

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; B_2, B_3 — второй, третий байты команды;

Z/NZ, C/NC, PE/PO, P/M (знак S), C′ — признаки результата;

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

[ ] — адрес ячейки памяти

Список команд микропроцессора 8080 приведен в табл. П2, П3. В табл. П2 представлены команды, не воздействующие на флаги процессора. 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. П3 представлены команды, при выполнении которых признаки результата (флаги) модифицируются в соответствии с значением результата. Таблицы П2 и П3 отличаются друг от друга наличием (табл. П3) или отсутствием колонки «признаки результата» (табл. П2).

Таблица П2

Название команды	Мнемоника команды	Код операции	Параметры команды б	ц	т	Описание команды
1	2	3	4			5
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 B_2	2	2	7	$B_2 \rightarrow (R_D)$
	MVI M,Data8	O66 B_2	2	3	10	$B_2 \rightarrow (M)$
Загрузка пары регистров непосредственная	LXI B,Data16	001 B_2 B_3	3	3	10	$B_2 \rightarrow (C)$ $B_3 \rightarrow (B)$
	LXI D,Data16	021 B_2 B_3	3	3	10	$B_2 \rightarrow (E)$ $B_3 \rightarrow (D)$
	LXI H,Data16	041 B_2 B_3	3	3	10	$B_2 \rightarrow (L)$ $B_3 \rightarrow (H)$
	LXISP,Data16	061 B_2 B_3	3	3	10	B_2 $B_3 \rightarrow (SP)$
Команды пересылок данных в аккумулятор и из аккумулятора						
Загрузка A (прямая)	LDA addr	072 B_2 B_3	3	4	13	$[(B_2)(B_3)] \rightarrow (A)$
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 B_2 B_3	3	4	13	$(A) \rightarrow [(B_2)(B_3)]$
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 B_2	2	3	10	[port] → (A)
Вывод	OUT port	323 B_2	2	3	10	(A) → [port]
Команды обмена, загрузки и запоминания содержимого регистровой пары HL						
Обменять	XCHG	353	1	1	4	(H) ↔ (D) (L) ↔ (E)
	XTHL	343	1	5	18	(L) ↔ ([SP]) (H) ↔ ([SP+1])
Загрузка (прямая)	LHLD addr	052 B_2 B_3	3	5	16	$[(B_2)(B_3)] \rightarrow$ (L) $[(B_2+1)(B_3)] \rightarrow$(H)
Запоминание (прямое)	SHLD addr	042 B_2 B_3	3	5	16	(L)] → $[(B_2)(B_3)$ (H)→$[(B_2+1)(B_3)]$
Загрузка указателя стека	SPHL	371	1	1	5	(H)(L) → (SP)
Загрузка счетчика команд	PCHL	351	1	1	5	(H)(L) → (PC)
Команды загрузки в стек и извлечения из стека						
Загрузка стека	PUSH B	305	1	3	11	(B) → ([SP-1]) (C) → ([SP-2]) (SP):= (SP)-2
	PUSH D	325	1	3	11	(D) → ([SP-1]) (E) → ([SP-2]) (SP):= (SP)-2
	PUSH H	345	1	3	11	(H) → ([SP-1]) (L) → ([SP-2]) (SP):= (SP)-2
	PUSH PSW	365	1	3	11	(A) → ([SP-1]) (F) → ([SP-2]) (SP):= (SP)-2
Извлечение из стека	POP B	301	1	3	10	([SP]) → (C) ([SP+1]) → (B) (SP):= (SP)+2
	POP D	321	1	3	10	([SP]) → (E) ([SP+1]) → (D) (SP):= (SP)+2
	POP H	341	1	3	10	([SP]) → (L) ([SP+1]) → (H) (SP):= (SP)+2
	POP PSW	361	1	3	10	([SP]) → (F) ([SP+1]) → (A) (SP):= (SP)+2

Продолжение таблицы П2

1	2	3	4			5
Команды безусловной передачи управления						
Безусловный переход	JMP addr	303 B_2 B_3	3	3	10	B_2 B_3→ (PC)
Безусловный вызов подпрограммы	CALL addr	315 B_2 B_3	3	5	17	(PC)→[(SP –2)(SP) –1] (SP):=(SP) –2 B_2 B_3→ (PC)
Безусловный возврат из подпрограммы	RET	311	1	3	10	[(SP) (SP) +1] → (PC) (SP):=(SP) +2
Загрузка счетчика команд (множественное ветвление)	PCHL	351	1	1	5	(H)(L) → (PC)
Вызов прерывающей программы	RST N	3N7	1	3	11	(PC)→[(SP –2)(SP) –1] (SP):=(SP) –2 (PC):=000 0N0
Команды условного перехода						
Переход по нулевому значению флага Z (ненулевому результату)	JNZ addr	302 B_2 B_3	3	3	10	Переход по условию, заданному в команде. Если условие выполняется, то осуществляется переход к команде, размещенной по адресу B_2 B_3 B_3 B_2 → (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 B_2 B_3	3	3	10	
Переход по нулевому значению флага C (при отсутствии переноса)	JNC addr	322 B_2 B_3	3	3	10	
Переход по единичному значению флага C (при наличии переноса)	JC addr	332 B_2 B_3	3	3	10	
Переход по нулевому значению флага четности P (при отсутствии четности)	JPO addr	342 B_2 B_3	3	3	10	
Переход по единичному значению флага четности P (при наличии четности)	JPE addr	352 B_2 B_3	3	3	10	
Переход по нулевому значению флага знака S, (положительное значение)	JP addr	362 B_2 B_3	3	3	10	
Переход по единичному значению флага знака S, (отрицательное значение)	JM addr	372 B_2 B_3	3	3	10	
Команды условного вызова подпрограмм						
Вызов подпрограммы при Z = 0	CNZ addr	304 B_2 B_3	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 B_2 B_3 B_2 B_3 → (PC). Если условие не выполняется, то осуществляется переход к следующей по порядку команде (PC):= (PC) +3
Вызов подпрограммы при Z = 1	CZ addr	314 B_2 B_3	3	3/5	11/17	
Вызов подпрограммы при C = 0	CNC addr	324 B_2 B_3	3	3/5	11/17	
Вызов подпрограммы при C = 1	CC addr	334 B_2 B_3	3	3/5	11/17	
Вызов подпрограммы при отсутствии четности P=0	CPO addr	344 B_2 B_3	3	3/5	11/17	
Вызов подпрограммы при наличии четности P=1	CPE addr	354 B_2 B_3	3	3/5	11/17	
Вызов подпрограммы при положительном результате (S = 0)	CP addr	364 B_2 B_3	3	3/5	11/17	
Вызов подпрограммы при отрицательном результате (S = 1)	CM addr	374 B_2 B_3	3	3/5	11/17	

Продолжение таблицы П2

1	2	3	4			5
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] → (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	
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
Запрещение 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 П3

Название команды	Мнемоника команды	Код операции	Признаки (флаги) результата s z c' p c					Параметры команды б ц т			Описание команды
1	2	3	4					5			6
Команды арифметических операций											
Сложение	ADD R_S	20S	+	+	+	+	+	1	1	4	(A) + (R_S) → (A)
	ADD M	206	+	+	+	+	+	1	2	7	(A) + (M) → (A)
	ADI Data8	306 B_2	+	+	+	+	+	2	2	7	(A) + B_2 → (A)
Сложение с переносом	ADC R_S	21S	+	+	+	+	+	1	1	4	(A)+(R_S)+ C→ (A)
	ADC M	216	+	+	+	+	+	1	2	7	(A)+(M)+ C → (A)
	ACI Data8	316 B_2	+	+	+	+	+	2	2	7	(A)+ B_2 + C → (A)
Вычитание	SUB R_S	22S	+	+	+	+	+	1	1	4	(A) - (R_S) → (A)
	SUB M	226	+	+	+	+	+	1	2	7	(A) - (M) → (A)
	SUI Data8	326 B_2	+	+	+	+	+	2	2	7	(A) – B_2 → (A)
Вычитание с заемом	SBB R_S	23S	+	+	+	+	+	1	1	4	(A) - (R_S) - C→(A)
	SBB M	236	+	+	+	+	+	1	2	7	(A) - (M) - C→ (A)
	SBI Data8	336 B_2	+	+	+	+	+	2	2	7	(A) - B_2 - C → (A)
Сравнение (неразрушающее	CMP R_S	27S	+	+	+	+	+	1	1	4	(A) - (R_S)
	CMP M	276	+	+	+	+	+	1	2	7	(A) - (M)

вычитание)	CPI Data8	376 B_2	+	+	+	+	+	2	2	7	(A) - B_2

Продолжение таблицы П3

Название команды	Мнемоника команды	Код операции	Признаки (флаги) результата s	z	c'	p	c	Параметры команды б	ц	т	Описание команды
1	2	3	4					5			6
Инкремент (увеличение содержимого адресуемого источника на 1)	INR R_D	0D4	+	+	+	+	-	1	1	5	(R_D) + 1 → (R_D)
	INR M	064	+	+	+	+	-	1	3	10	(M) + 1 → (M)
	INX B	003	-	-	-	-	-	1	1	5	(B)(C) + 1→(B)(C)
	INX D	023	-	-	-	-	-	1	1	5	(D)(E) + 1→(D)(E)
	INX H	043	-	-	-	-	-	1	1	5	(H)(L) + 1→(H)(L)
	INX SP	063	-	-	-	-	-	1	1	5	(SP) + 1 →(SP)
Декремент (уменьшение содержимого адресуемого источника на 1)	DCR R_D	0D5	+	+	+	+	-	1	1	5	(R_D) – 1 → (R_D)
	DCR M	065	+	+	+	+	-	1	3	10	(M) – 1 → (M)
	DCX B	013	-	-	-	-	-	1	1	5	(B)(C) - 1 →(B)(C)
	DCX D	033	-	-	-	-	-	1	1	5	(D)(E) - 1 →(D)(E)
	DCX H	053	-	-	-	-	-	1	1	5	(H)(L) - 1 →(H)(L)
	DCX SP	073	-	-	-	-	-	1	1	5	(SP) - 1 →(SP)
Двойное сложение	DAD B	011	-	-	-	-	+	1	3	10	HL + BC → HL
	DAD D	031	-	-	-	-	+	1	3	10	HL + DE → HL
	DAD H	051	-	-	-	-	+	1	3	10	HL + HL → HL
	DAD SP	071	-	-	-	-	+	1	3	10	HL + SP → HL
Десят. коррекция	DAA	047	+	+	+	+	+	1	1	4	
Команды логических операций											
Логическое умножение	ANA R_S	24S	+	+	-	+	0	1	1	4	(A) ^ R_S → (A)
	ANA M	246	+	+	-	+	0	1	2	7	(A) ^ M → (A)
	ANI Data8	346 B_2	+	+	-	+	0	2	2	7	(A) ^ B_2→ (A)
Исключающее ИЛИ (сложение по mod 2)	XRA R_S	25S	+	+	-	+	0	1	1	4	(A) ⊕ R_S → (A)
	XRA M	256	+	+	-	+	0	1	2	7	(A) ⊕ M → (A)
	XRI Data8	356 B_2	+	+	-	+	0	2	2	7	(A) ⊕ B_2 → (A)
Логическое сложение	ORA R_S	26S	+	+	-	+	0	1	1	4	(A) ∨ R_S → (A)
	ORA M	266	+	+	-	+	0	1	2	7	(A) ∨ M → (A)
	ORI Data8	366 B_2	+	+	-	+	0	2	2	7	(A) ∨ B_2 → (A)
Инвертирование A	CMA	057	-	-	-	-	-	1	1	4	→ (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 C											
Установка переноса	STC	067	-	-	-	-	+	1	1	4	1 → C
Инверсия переноса	CMC	077	-	-	-	-	+	1	1	4	$\overline{C}$ → C

Приложение 2

Типы машинных циклов МП КР580ВМ80А и состояние управляющих сигналов байта состояния для различных типов машинных циклов приведено в табл. П2.1.

Таблица П2.1

Машинный цикл	Кодирование управляющих сигналов байта состояния							
	B_7 MEMR	B_6 INP	B_5 M1	B_4 OUT	B_3 HLTA	B_2 STACK	B_1 WO	B_0 INTA
M1 Выборка кода операции	1	0	1	0	0	0	1	0
M2 Считывание из памяти	1	0	0	0	0	0	1	0
M3 Запись в память	0	0	0	0	0	0	0	0
M4 Считывание из стека	1	0	0	0	0	1	1	0
M5 Запись в стек	0	0	0	0	0	1	0	0
M6 Ввод (считывание ВУ)	0	1	0	0	0	0	1	0
M7 Вывод (запись ВУ)	0	0	0	1	0	0	0	0
M8 Прерывание	0	0	1	0	0	0	1	1
M9 Останов	1	0	0	0	1	0	1	0
M10 Прерывание при останове	0	0	1	0	1	0	1	1

Назначение отдельных бит B_i байта состояния [2]:

B_0 (INTA) — подтверждение прерывания (ППР). Используется для ввода в регистр команд МП команды RST из устройства, запрашивающего прерывание.

B_1 (WO) — запись/вывод (ВМП). Идентифицирует МП как источник информации (WO=0) при записи в память/выводе или как получатель информации (WO=1) при чтении из памяти/вводе.

B_2 (STACK) — (СТЕК). Источником адреса на шине адреса является указатель стека.

B_3 (HLTA) — подтверждение останова (ПОСТ). МП находится в состоянии останова.

B_4 (OUT) — вывод (ЗПВВ). На шине адреса установлен адрес порта вывода.

B_5 (M1) — машинный цикл выборки команды M1. Определяет выборку из программной памяти кода операции (первого байта команды).

B_6 (INP) — ввод (ЧТВВ). На шине адреса установлен адрес порта ввода.

B_7 (MEMR) — считывание из памяти (ЧТП). На шине адреса установлен адрес памяти.

Приложение 3

Назначение элементов управления режимами работы лабораторного стенда DiLaB 8080.

Блок PUSH BUTTENS:

Кнопка LEFT обеспечивает переключение режимов работы: из режима WORK S в режим DMA и обратно при повторном нажатии;

Кнопка RIGHT используется для формирования сигнала «ПУСК» машинного цикла при выполнении команд в пошаговом режиме WORK S;

Кнопки UP и DOWN обеспечивают последовательный перебор адресов в режиме доступа к ячейкам памяти в режиме DMA Кнопка UP инкрементирует адрес, а кнопка DOWN – декрементирует его.

Блок переключателей SWITCHES:

Переключатели SWITCHES (SW1 – SW8) в режиме DMA используются для модификации кода старшего и младшего байтов адреса, а также кода 8-битных данных. В режиме DMA +/- переключатели SWITCHES (SW1 – SW8) задают двоичный код вводимых 8-битных данных.

16-кнопочная клавиатура (KEYPAD). Отдельные кнопки клавиатуры используются при формировании сигналов управления режимами работы:

Кнопки A, B, D используются при загрузке двоичного кода вводимых данных в буферный регистр (адреса или данных): кнопка «A» при вводе старшего байта адреса, кнопка «B» - младшего байт адреса, кнопка «D» - непосредственно 8 битных данных.

Кнопка «C» предназначена для запуска программы в режиме непрерывного выполнения. Кнопка «C» активизируется только в режиме WORK S. При нажатии кнопки «C» осуществляется автоматический запуск программы на выполнение и она выполняется полностью. По завершению программы (после выполнения последней команды HLT) процессор переводится в состояние «ОСТАНОВА», в котором шины адреса и данных МП устанавливаются в состояние высокого сопротивления. Возврат в основной режим пошагового выполнения команд WORK S осуществляется при нажатии кнопки «*» - RESET.

Кнопка «*» - инициализация микропроцессорной системы. При нажатии кнопки «*» счетчик команд МП 8080 устанавливается в «0», триггеры разрешения прерывания и подтверждения захвата сбрасываются, система переводится в режим пошагового выполнения команд WORK S: процессор ожидает поступления сигнала RIGHT (ПУСК) для выполнения команды, записанной по нулевому адресу.

Кнопка «0» предназначена для переключения в режим ВЫВОДА параллельной информации. Повторное нажатие кнопки «0» переключает микротренажер в режим пошагового выполнения программы по машинным циклам.

Кнопка «1» предназначена для переключения в режим пошагового выполнения программы по командным циклам.

Кнопка «#» предназначена для переключения в режим отображения с использованием блока статической индикации.

Приложение 4

Кодирование управляющих полей микрокоманды МП К1804

Выполняемая в АЛУ операция определяется набором управляющих сигналов, формируемых дешифратором блока управления ЦПЭ из сигналов I5 I4 I3 микрокоманды:

Микрокод операции АЛУ			Реализуемая операция
I5	I4	I3	
0	0	0	R + S + C0
0	0	1	S - R – 1 + C0
0	1	0	R - S – 1 + C0
0	1	1	R ∨ S
1	0	0	R ^ S
1	0	1	$\overline{R}$ ^ S
1	1	0	R⊕S
1	1	1	$\overline{R \oplus S}$

Выбор источников операндов АЛУ, подключаемых к входам R и S, осуществляет мультиплексор, управляемый полем I2 I1 I0 кода микрокоманды:

Микрокод выбора источников операндов			Источник операнда	
I2	I1	I0	R	S
0	0	0	A	Q
0	0	1	A	B
0	1	0	0	Q
0	1	1	0	B
1	0	0	0	A
1	0	1	D	A
1	1	0	D	Q
1	1	1	D	0

П р и м е ч а н и я: A, B — содержимое РОНов, определяемых адресами A и B; Q — содержимое дополнительного регистра Q; D — значение данных на входе ЦПЭ; 0 — константа «логический нуль».

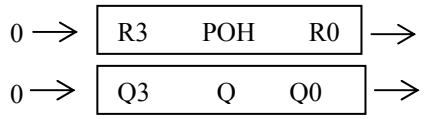
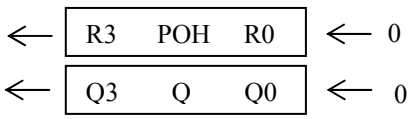
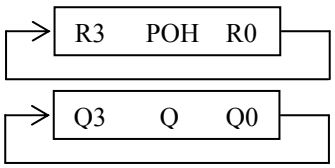
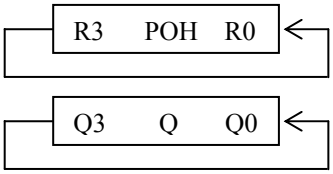
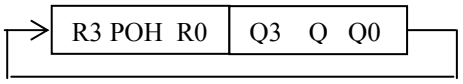
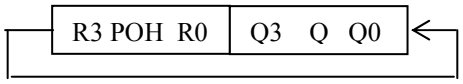
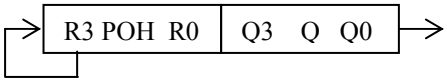
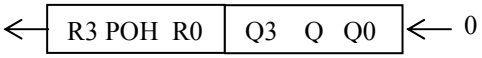
Приемник результата выполненной в АЛУ операции со сдвигом или без сдвига кодируется разрядами I8 I7 I6 микрокоманды.

Двоичный код, задающий приемник результата			**Регистровое ЗУ**		**Регистр Q**		**Выход Y ЦПЭ**
I8	**I7**	**I6**	**Сдвиг**	**Загрузка**	**Сдвиг**	**Загрузка**	
0	**0**	**0**	**–**	**–**	**–**	**F → Q**	**F**
0	**0**	**1**	**–**	**–**	**–**	**–**	**F**
0	**1**	**0**	**–**	**F → B**	**–**	**–**	**A**
0	**1**	**1**	**–**	**F → B**	**–**	**–**	**F**
1	**0**	**0**	**Вправо**	**F/2 → B**	**Вправо**	**Q/2 → Q**	**F**
1	**0**	**1**	**Вправо**	**F/2 → B**		**–**	**F**
1	**1**	**0**	**Влево**	**2F → B**	**Влево**	**2Q → Q**	**F**
1	**1**	**1**	**Влево**	**2F → B**		**–**	**F**

Примечания:

F — выход АЛУ; – — отсутствие операции сдвига или загрузки; A — выход регистра, адресуемого полем A; B, Q — адрес приемника результата (имя регистра, адресуемого полем B или регистр Q соответственно);.

Тип сдвиговой операции и направление сдвига на входе регистрового ЗУ и регистра Q задается старшими разрядами 4-й и 5-й тетрад микрокоманды, обозначенными как MS1 и MS2. Сигналы MS2 и MS1 управляют внешним мультиплексором сдвига.

Код		Сдвиг вправо	Сдвиг влево
MS2	MS1		
0	0	Сдвиг 4-разрядн. слова с вводом лог.0 в старшие разряды РОН и регистр Q 	Сдвиг 4-разрядного слова с вводом лог.0 в младшие разряды РОН и регистр Q 
0	1	Циклический сдвиг 4-разрядного слова 	Циклический сдвиг 4-разрядного слова 
1	0	Циклический сдвиг 8-разрядного слова 	Циклический сдвиг 8-разрядного слова 
1	1	Арифметический сдвиг 8-разрядного слова влево с вводом знакового разряда в старший разряд РОН 	Арифметический сдвиг 8-разрядного слова с вводом лог.0 в младший разряд регистра Q 

Типы возможных переходов в зависимости от кода инструкции микрокоманды представлены в таблице:

Код инструкции перехода				Тип перехода
P3	P2	P1	P0	
0	0	0	0	Переход на адрес из регистра микрокоманд, если Z=0
0	0	0	1	Переход на адрес из регистра микрокоманд
0	0	1	0	Продолжить (переход на следующий адрес)
0	0	1	1	Переход на адрес, формируемый клавишным регистром адреса (переход на адрес вектора)
0	1	0	0	Переход к подпрограмме, если Z=0
0	1	0	1	Переход к подпрограмме
0	1	1	0	Возврат из подпрограммы
0	1	1	1	Переход по стеку
1	0	0	0	Окончить цикл и вытолкнуть из стека, если Z=1
1	0	0	1	Загрузить стек и продолжить
1	0	1	0	Вытолкнуть из стека и продолжить
1	0	1	1	Окончить цикл и вытолкнуть из стека, если C4=1
1	1	0	0	Переход на адрес из регистра микрокоманд, если Z=1
1	1	0	1	Переход на адрес из регистра микрокоманд, если F3=1
1	1	1	0	Переход на адрес из регистра микрокоманд, если OVR=1
1	1	1	1	Переход на адрес из регистра микрокоманд, если C4=1

Приложение 5.

Назначение элементов управления режимами работы лабораторного стенда DiLab 1804

Блок PUSH BUTTENS:

Кнопка UP обеспечивает переключение стенда в режим «ЗАГРУЗКА» (при этом на левом индикаторе LED MONITOR отобразится буква **З**);

Кнопка DOWN обеспечивает переключение стенда в режим «РАБОТА» (при этом на левом индикаторе LED MONITOR отобразится буква **Р**);

Кнопка RIGHT используется для формирования сигнала записи адреса микропрограммной памяти (при этом номер адреса отображается в десятичной форме на трех правых индикаторах LED MONITOR);

Кнопка LEFT используется в режиме «РАБОТА» для пошагового выполнения микрокоманд из памяти микропрограмм;

Блок переключателей SWITCHES:

Переключатели SWITCHES в режиме «ЗАГРУЗКА» используются для установки номера тетрады микрокоманды (переключатели SW1 – SW4) и установки четырехразрядного кода данных для записи в выбранную тетраду (переключатели SW5 – SW8). При этом по показаниям светодиодов LED INDICATORS можно контролировать правильность набора тетрады и ее содержимого.

16-кнопочная клавиатура (KEYPAD) используется в режиме «ЗАГРУЗКА» для набора десятичного адреса микропрограммной памяти; набору трехразрядного адреса от 000 до 255 должен предварять набор кнопки «*». Набранный адрес отображается на индикаторах LED MONITOR.

Приложение 6

Примеры и варианты заданий лабораторного практикума

Приобретение практических навыков программирования прикладных задач с использованием систем команд (микрокоманд) конкретных микропроцессоров является одним из важнейших аспектов подготовки специалистов в области микропроцессорных систем. Разработка прикладных программ в большинстве случаев не поддается формализации и в основном опирается на опыт и здравый смысл. Практические навыки решения прикладных задач в лаборатории приобретаются при разработке программ по индивидуальным заданиям и анализе известных пользовательских программ. Варианты индивидуальных заданий и некоторые особенности алгоритмов предлагаемых для решения прикладных задач рассматриваются ниже. Программы индивидуальных заданий реализуются на учебных лабораторных стендах, представляющих собой микропроцессорные системы с определенным типом микропроцессора.

Варианты индивидуальных заданий

1. Нахождение минимального (максимального) числа в массиве;
2. Упорядочение элементов массива по возрастанию (убыванию);
3. Нахождение медианы массива элементов;
4. Определение моды массива элементов;
5. Вычисление элементов матрицы, полученной в результате суммирования двух исходных матриц;
6. Табличное преобразование кода;
7. Определение максимального двоичного значения в массиве из n чисел, представленных в коде Грея;
8. Транспонирование матрицы $n \times n$;
9. Умножение «младшими разрядами вперед» двух n-разрядных чисел ($n = 4, 8$);
10. Умножение «старшими разрядами вперед» двух n-разрядных чисел ($n = 4, 8$);
11. Деление «с восстановлением остатка» двух 8-разрядных чисел с формированием 8-разрядного частного;
12. Деление «без восстановления остатка» двух 8-разрядных чисел с формированием 8-разрядного частного;
13. Определение n-го члена и суммы n членов ряда Фибоначчи;
14. Статическая индикация элементов массива чисел (матрицы, сомножителей и произведения, делимого, делителя и частного);
15. Динамическая индикация элементов массива чисел (матрицы, сомножителей и произведения, делимого, делителя и частного);
16. Управление макетом светофора;
17. Программный секундомер;
18. Реализация «бегущей» строки заданного текста.
19. Поиск максимального (минимального, срединного) из трех чисел методом «дерева» сравнений;

20. Определение числа нулей (единиц) в байте;
21. Определение числа совпадающих (несовпадающих) разрядов в двух байтах;
22. Определение числа переходов из 1 в 0 и из 0 в 1 в байте данных;
23. Определение числа переходов из 1 в 0 (из 0 в 1) в байте данных;
24. Определение максимальной последовательности нулей (единиц) в байте;
25. Определение делимости на три 8-разрядного числа;
26. Преобразование двоичных чисел в двоично-десятичные;
27. Преобразование двоично-десятичных чисел в двоичные.

При решении любой задачи необходимо:

1. Проанализировать задание и выбрать аппаратные и программные средства для решения. При разработке программ лабораторного практикума на этом этапе определяется состав операционной части МП, требуемый для решения задачи (выполняются соответствующие назначения регистров блока РОН — кто за что отвечает).
2. Составить алгоритм решения. Уровень проработки должен быть таким, чтобы на его основе можно было легко написать программу.
3. В соответствии с разработанным алгоритмом написать программу (микропрограмму) решения задачи с учетом особенностей конкретного МП.
4. Выполнить программирование задачи на стенде.
5. Провести настройку и тестирование программы.
6. Составить отчет, включающий:
   – схему алгоритма;
   – текст программы с необходимыми комментариями;
   – результаты тестирования выполненной программы;
   – выводы, отражающие особенности реализации при решении задачи с помощью конкретного МП.

Пояснения и рекомендации к программам индивидуальных заданий

Задания разбиты на группы в соответствии с назначением и особенностями реализуемых алгоритмов.

Обработка массивов чисел (структур данных)

Под структурами данных (массивами) понимают наборы некоторым образом организованных данных. Различают простые структуры данных, в которых каждый элемент массива определяется своим номером (индексом), и сложные структуры, например двухсвязные списки, элементы которых вместе с данными должны содержать указатели преемника и предшественника. Программисту необходимо знать наиболее распространенные структуры данных и способы хранения их в памяти ВМ. Правильный выбор структуры данных позволяет уменьшить объем памяти и увеличить производительность системы. Чаще применяются простые структуры данных. Наиболее простой и распространенной структурой данных является *одномерный массив.* Он представляет собой конечный набор элементов одинаковой длины, который

размещается в смежных ячейках памяти, начиная с некоторого начального адреса. Положение элемента в массиве определяется его индексом. Адрес элемента массива равен сумме базового адреса и индекса, умноженного на длину элемента в байтах.

Важными характеристиками массива элементов являются **медиана** и **мода**. Медианой массива, содержащего нечетное число элементов, называется элемент *m* этого массива, характеризующийся тем, что половина оставшихся элементов массива больше или равна *m*, а другая половина меньше или равна *m*. Если число элементов массива четное, то медиана массива есть среднее двух элементов *m1* и *m2* такое, что половина элементов массива больше или равна *m1* и *m2*, а другая половина меньше или равна *m1* и *m2*.

Модой массива называется число *m*, которое встречается в массиве наиболее часто. Если в массиве имеется несколько часто встречающихся чисел и число их вхождений совпадает, то считается, что массив не имеет моды.

Обработка массивов осуществляется циклическими программами, состоящими из трех частей: инициализация, обращение к текущему элементу и переход к следующему элементу, проверка окончания цикла. В программах обязательно должны использоваться два регистра: регистр с адресом текущего элемента и регистр-счетчик, фиксирующий окончание цикла.

Массив не обязательно должен быть линейным набором элементов. Он может быть и многомерным. *Двумерный массив* представляет набор данных, в котором доступ к любому из элементов осуществляется по двум индексам — номеру строки n_r и номеру столбца n_c. Пример двумерного массива показан на рис. П6.1.

	Столбец 1	Столбец 2	Столбец 3	Столбец 4
Строка 1	A(1,1)	A(1,2)	A(1,3)	A(1,4)
Строка 2	A(2,1)	A(2,2)	A(2,3)	A(2,4)
Строка 3	A(3,1)	A(3,2)	A(3,3)	A(3,4)
Строка 4	A(4,1)	A(4,2)	A(4,3)	A(4,4)

Рис. П6.1. Двумерный массив

Двумерный массив хорошо иллюстрирует различие между *физическим* и *логическим* представлением данных. Двумерный массив представляет собой логическую структуру данных, которая удобна для программирования и решения задач. Например, с помощью такого массива удобно представлять матрицы элементов. Физическое представление данных отличается от логического. Это связано с тем, что память, в которой хранится массив, имеет линейную организацию, т. е. является одномерным массивом. Для извлечения какого-либо элемента из памяти используется один адрес. Для представления двумерного массива в памяти используется либо отображение по столбцам (рис. П6.2а), либо отображение по строкам (рис. П6.2б). При отображении по столбцам первый столбец массива занимает первую группу ячеек памяти, второй столбец занимает следующую группу и т. д. При запоминании

элементов, упорядоченных по строкам, за элементами первой строки следуют элементы второй строки и т. д.

A(1,1)
A(2,1)
A(3,1)
A(4,1)
A(1,2)
A(2,2)
A(3,2)
A(4,2)
A(1,3)
A(2,3)
A(3,3)
A(4,3)
A(1,4)
A(2,4)
A(3,4)
A(4,4)

а)

A(1,1)
A(1,2)
A(1,3)
A(1,4)
A(2,1)
A(2,2)
A(2,3)
A(2,4)
A(3,1)
A(3,2)
A(3,3)
A(3,4)
A(4,1)
A(4,2)
A(4,3)
A(4,4)

б)

Рис. П6.2. Отображение двумерного массива в памяти по столбцам (а) и по строкам (б)

Как и в одномерном массиве, в двумерном массиве положение элемента определяется его индексом. Адрес элемента массива равен сумме базового адреса и общего индекса IND, который зависит от способа размещения массива в памяти:

IND = $i * n_r + j$, если массив хранится по строкам,

IND = $i + j * n_c$, если массив хранится по столбцам,

здесь i — номер строки, j — номер столбца, n_r и n_c — число строк и столбцов.

Обработка двумерных массивов выполняется подпрограммами, содержащими два цикла. Во внешнем цикле производится инкремент i, а во внутреннем (вложенном) — инкремент j.

Типовыми задачами при работе с одномерными массивами данных являются:

– нахождение минимального (максимального) числа в массиве;

– упорядочение элементов массива по возрастанию (убыванию);

– нахождение медианы массива элементов;

– определение моды массива элементов;

– вычисление элементов матрицы, полученной в результате суммирования двух исходных матриц;

– табличное преобразование кода.

Примером обработки двумерного массива является задача перестановки элементов массива при транспонировании матрицы заданного размера.

П6.1. Выбор одного из m чисел по заданному критерию. Критерием выбора может быть: максимальное число, минимальное, «медианное», наибольшее четное и др.

Общим методом решения данной задачи при работе с одномерными массивами чисел является метод обменной сортировки, осуществляющий упорядочение элементов массива в возрастающем или убывающем порядке. Существует большое разнообразие методов сортировки. Простейшим из них является метод пузырьковой сортировки. Сортировка заключается в последовательном просмотре элементов массива, сравнении значений соседних элементов и обмене этих элементов, если они расположены в порядке, отличном от требуемого. После первого просмотра массива наибольший (наименьший) элемент оказывается в нужной позиции. Просмотры повторяются до полного упорядочения всех элементов массива. Рассмотренный метод сортировки называется «методом пузырька» потому, что при упорядочении элементы исходной последовательности как бы «всплывают» (медленно перемещаются) на нужную позицию. В общем случае для сортировки методом пузырька требуется $(n-1)$ просмотров массива, в каждом из которых выполняются $(n-1)$ сравнение. После завершения сортировки выбор элемента по заданному критерию не представляет труда.

Большое число сравнений, характерное для классической пузырьковой сортировки является недостатком этого метода, поскольку требует значительных временных затрат. Известно несколько способов, позволяющих уменьшить время сортировки. Например, можно учесть, что после очередного i-го просмотра массива все элементы в позициях $n-i+1$ оказываются упорядоченными и просмотр массива можно начинать $(n-i)$-го элемента. Отмеченное свойство пузырьковой сортировки позволяет уменьшить общее число сравнений и, благодаря этому, ускорить процесс упорядочения элементов массива.

Число сравнений при сортировке можно также уменьшить, если учесть, что в большинстве случаев массив оказывается отсортированным после меньшего, чем $n-i$, числа просмотров и последующие просмотры оказываются ненужными. Для исключения ненужных просмотров используют следующий прием. В программу вводят переменную — специальный указатель обмена, с помощью которого фиксируют наличие хотя бы одной перестановки в просмотре. Значение указателя обмена анализируется после окончания просмотра массива. При ненулевом указателе последний сбрасывается, и весь массив просматривается вновь. Сортировка заканчивается, когда при очередном просмотре массива обмены не фиксируются.

Существенного сокращения числа сравнений при решении задачи выбора одного из m чисел по заданному критерию можно достичь, используя алгоритм «дерева» сравнений и переходов. На рис. П6.3 представлен достаточно очевидный алгоритм «дерева» сравнений для массива из трех элементов. При его реализации выполняется только ограниченное число операций сравнения, вследствие чего этот алгоритм является значительно более быстродействующим по сравнению с алгоритмами сортировки. Для нахождения медианы в массиве из трех элементов требуется пять операций сравнения, при этом медиана для конкретного набора элементов определяется максимум за три сравнения.

Недостатком данного метода является его неэффективность по затратам памяти с ростом размерности массива. По этой причине метод «дерева» сравнений используют только для массивов малой размерности, как правило, не более чем из трех элементов.

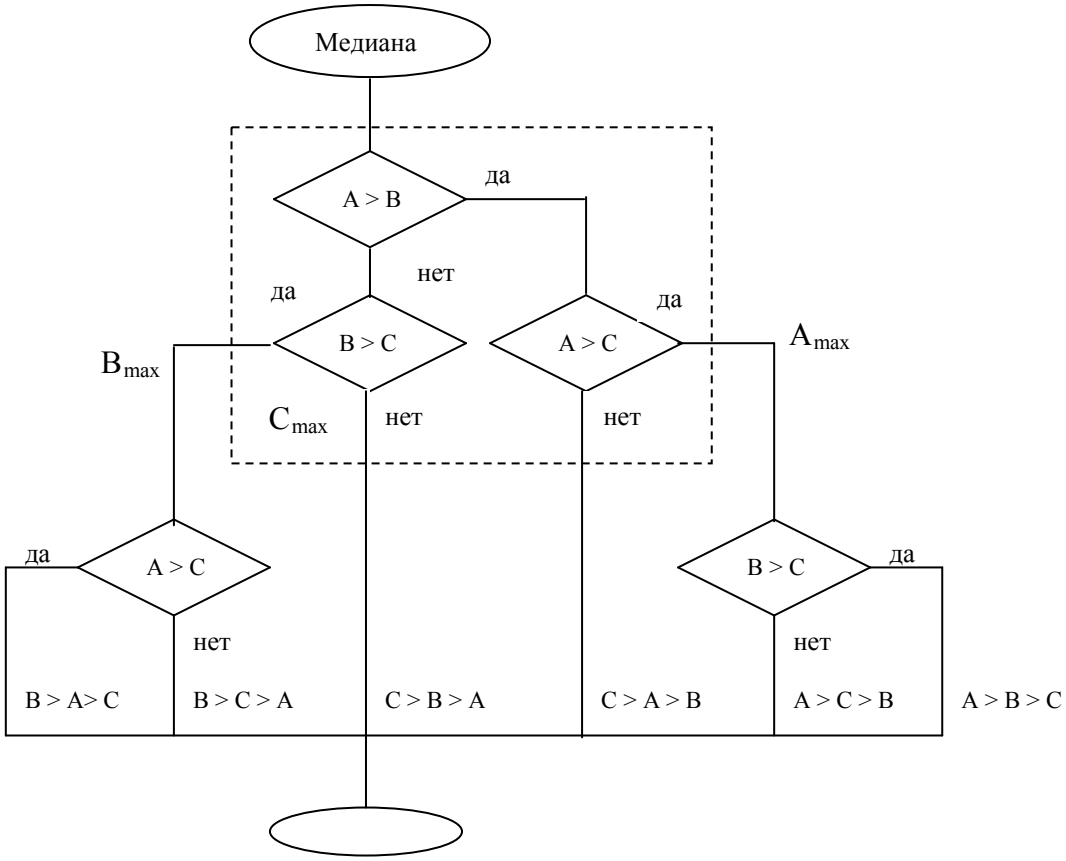


Рис. П6.3 Дерево сравнений для поиска элемента массива из трех чисел по заданному критерию.

Алгоритм поиска максимального/минимального числа естественно проще. В частности для поиска максимального значения из трех элементов (алгоритм поиска на рис. П6.3 выделен пунктиром) требуется не более двух сравнений.

Более простым в реализации может оказаться алгоритм поиска числа по заданному критерию, представленный на рис. П.6.4. По сравнению с алгоритмом классического «дерева» сравнений этот алгоритм требует меньшего числа различных сравнений (всего двух), но является менее быстродействующим, поскольку при его реализации возможны дополнительные перестановки элементов массива.

П6.2. Вывод на семисегментные индикаторы последовательности буквенных символов или n-разрядных чисел в заданной системе счисления

Индикация отображаемой информации осуществляется с помощью блоков статической или динамической индикации, подключаемых к стенду-тренажеру ТУМ1 с микропроцессором КР580ВМ80А (см. п. 4.1.4).

При разработке программы «индикации» необходимо решить, по крайней мере, две практические задачи:

– преобразовать цифровые коды в заданной системе счисления (восьмеричной, десятичной или шестнадцатеричной) в семисегментные;

– реализовать программные задержки, необходимые для обеспечения приемлемой или требуемой скорости при последовательном выводе визуальной информации.

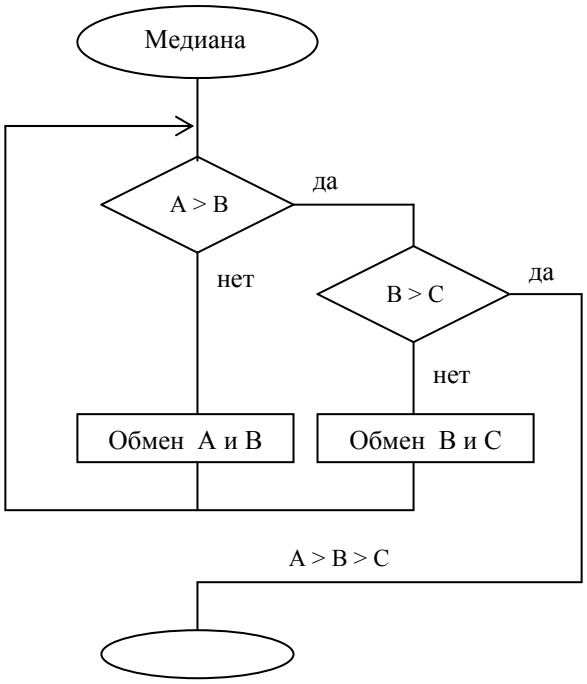


Рис. П6.4. Алгоритм поиска элемента массива из трех чисел по заданному критерию

Рассмотрим подходы к решению названных задач подробнее.

П.3.2.1. Табличное преобразование кода.

При индикации десятичных или восьмеричных цифр, а также букв латинского и русского алфавитов с помощью семисегментных индикаторов на светоизлучающих диодах коды отображаемых буквенных символов и цифр предварительно должны быть преобразованы в 7-разрядный код индикатора. Простой связи между кодами индицируемых символов и их 7-разрядными кодами нет. Для преобразования кодов двоично-десятичных цифр (и восьмеричных цифр как подпространства десятичных) в семисегментные коды индикаторов удобно использовать табличное преобразование кодов (табл. П6.1). В каждой строке табл. П6.1 содержится семисегментный код BCD-цифры. Суть преобразования при определении семисегментного кода BCD-цифры заключается в использовании BCD кода цифры в качестве индекса таблицы.

Программа преобразования может быть реализована по следующему алгоритму:

– таблица с семисегментными кодами BCD-цифр размещается в произвольной области памяти по некоторому базовому адресу;

– формируется адрес строки таблицы с семисегментным кодом искомой BCD-цифры (индекс таблицы). Эта операция реализуется путем прибавления BCD-цифры к базовому адресу таблицы;

– выполняется считывание соответствующей строки таблицы с искомым семисегментным кодом BCD-цифры.

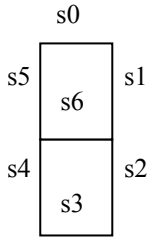


Таблица П6.1

BCD-цифра	7-разрядный код индицируемой цифры							
	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
0 0 0 0	0	0	1	1	1	1	1	1
0 0 0 1	0	0	0	0	0	1	1	0
0 0 1 0	0	1	0	1	1	0	1	1
0 0 1 1	0	1	0	0	1	1	1	1
0 1 0 0	0	1	1	0	0	1	1	0
0 1 0 1	0	1	1	0	1	1	0	1
0 1 1 0	0	1	1	1	1	1	0	1
0 1 1 1	0	0	0	0	0	1	1	1
1 0 0 0	0	1	1	1	1	1	1	1
1 0 0 1	0	1	1	0	1	1	1	1

П.6.2.2. Формирование программируемых временных задержек

Для формирования программируемых временных задержек обычно используют циклическую программу, выполняемую по следующему алгоритму.

В один из рабочих регистров МП загружают расчетное число, определяющее требуемое количество программных циклов. В каждом цикле программы кроме декремента содержимого рабочего регистра и проверки его значения никаких других полезных действий не выполняется. Формирование временной задержки завершается при нулевом значении рабочего регистра. Величина задержки определяется числом циклов, числом команд в цикле, и временем выполнения каждой команды. Наиболее длительная задержка формируется при загрузке в рабочий регистр числа 0. В этом случае задержка $\tau_{макс}$ равна

$$\tau_{макс} = N_{макс} \sum (n_k \tau_k),$$

где $N_{макс}$ — максимальное значение числа циклов, определяемое n-разрядным числом; n_k — число команд k-го типа в цикле, τ_k время выполнения команд k-го типа.

Для получения более длительных задержек необходимо использовать несколько рабочих регистров для хранения числа N, определяющего требуемое количество программных циклов.

Требуемая точность формирования временных задержек в большинстве случаев определяется решаемой задачей. В задачах «индикации» высокой точности формирования задержки τ не требуется. При решении других задач, в частности, при формировании единичных тиков времени в программном секундомере от точности «тика» зависит точность работы секундомера.

П6.2.3. Программный секундомер.

При реализации программного секундомера решается аналогичная задача преобразования кодов при выводе отметок времени на семисегментные индикаторы и формируются программируемые временные задержки. При этом в цикл временной задержки необходимо включать не только время формирования единичного тика, но и время, затрачиваемое на счет минут, секунд и десятых долей секунды и вывод текущего времени на семисегментные индикаторы.

П6.2.4. Вывод на световые индикаторы «бегущей строки» текста

Под строкой текста понимается последовательность индицируемых символов. «Бегущая строка» — это последовательность индицируемых символов, которые перемещаются по экрану светодиодного дисплея с заданной скоростью. В «бегущей строке» количество символов превышает число индикаторов. При программной реализации символ строки размещается в произвольной непрерывной области памяти. Для определения их местоположения используют указатели начального и конечного адресов символов. Направление движения текста в строке может осуществляться справа налево или в обратном направлении.

Перемещение текста организуется следующим образом. На индикаторы последовательно выводится фрагмент текста из нескольких символов, число которых определяется количеством светодиодных индикаторов. В течение программируемого интервала времени, достаточного для визуального наблюдения, этот фрагмент индицируется. После этого осуществляется инкремент указателя начала фрагмента и вывод на индикаторы очередного фрагмента текста из нескольких символов. Далее последовательно выводятся и индицируются следующие фрагменты текста. Число фрагментов соответствует числу выводимых символов. Для смены фрагмента достаточно изменить указатель начала фрагмента. Поскольку время вывода фрагмента на индикаторы составляет около 100 мкс, а время его индикации — порядка 1 с, у наблюдателя создается иллюзия непрерывной «бегущей строки». Рассмотренная последовательность действий оформляется в виде циклической процедуры, в которой рабочей частью цикла является последовательность команд, обеспечивающая вывод фрагмента текста и временную задержку для его индикации. Для того чтобы в начале вывода строки первый символ появлялся на правом индикаторе, а в конце вывода последний символ достигал левого индикатора и затем исчезал, текст дополняют нулевыми (неиндицируемыми) символами.

П6.3. Преобразование стандартного двоичного кода в двоичный код Грея и обратное преобразование

Для представления целых чисел, наряду со стандартным двоичным кодом, можно использовать двоичный код Грея, относящийся к классу рефлексивных (отраженных) кодов. При использовании кода Грея пара чисел, отличающаяся на единицу младшего разряда, представляется двоичными кодовыми значениями, которые различаются только в каком-либо одном двоичном разряде.

Алгоритм преобразования m-разрядного двоичного кода ($b_{m-1}\ b_m\ \ldots\ b_1\ b_0$) в m-разрядный код Грея ($g_{m-1}\ g_{m-2}\ \ldots\ g_1\ g_0$) можно представить совокупностью следующих выражений:

$$g_{m-1} = b_{m-1} \quad \text{и} \quad g_i = b_i \oplus b_{i+1} \quad (0 \le i \le m-1).$$

Нетрудно видеть, что для преобразования двоичного кода B в код Грея G достаточно выполнить логический сдвиг вправо числа B и сложить его по mod 2 с исходным значением B.

Алгоритм преобразования m-разрядного кода Грея ($g_{m-1}\ g_m\ \ldots\ g_1\ g_0$) в m-разрядный двоичный код ($b_{m-1}\ b_m\ \ldots\ b_1\ b_0$) представим совокупностью следующих выражений:

$$b_{m-1} = g_{m-1}$$

$$b_{m-2} = g_{m-2} \oplus g_{m-1}$$

$$\ldots$$

$$b_0 = g_0 \oplus g_1 \oplus \ldots g_{m-1}.$$

После несложных преобразований получим соотношения, в соответствии с которыми преобразование m-разрядного кода Грея в двоичный код не представляет труда:

$$b_{m-1} = g_{m-1};$$

$$b_{m-2} = b_{m-1} \oplus g_{m-2};$$

$$\ldots$$

$$b_0 = b_1 \oplus g_0.$$

Таким образом, преобразование кода Грея в двоичный код можно выполнить путем последовательного суммирования по mod 2 числа, полученного при логическом сдвиге вправо кода Грея, с числом, сформированным при предыдущем суммировании по mod 2.

П6.4. Умножение чисел с фиксированной точкой программным способом

«Простое» умножение, выполняемое путем последовательного суммирования множимого в соответствии со значением множителя, требует при реализации значительных временных затрат (до 2^n циклов суммирования, где n — разрядность множителя). На практике умножение реализуют методом последовательного суммирования частичных произведений (ΣЧП), образующихся в результате умножения множимого на соответствующий разряд множителя. Фактически умножение сводится к выполнению циклической процедуры из n циклов (по числу разрядов множителя). В каждом цикле осуществляются операции сдвига и сложения: сдвигается множитель и анализируется значение выдвигаемого бита. Если он равен 1, то выполняется сложение множимого с ΣЧП, если 0 — суммирование не происходит. При формировании ΣЧП можно суммировать либо сдвигаемое множимое и неподвижную ΣЧП, либо неподвижное множимое и сдвигаемую ΣЧП. Цикл

завершается сдвигом либо множимого при неподвижной ΣЧП, либо ΣЧП при неподвижном множимом. Умножение может начинаться с младшей цифры множителя (умножение младшими разрядами вперед) или со старшей (умножение старшими разрядами вперед). В зависимости от того, что сдвигается (множимое или ΣЧП) и в каком направлении осуществляется сдвиг, различают четыре способа умножения (табл. П6.2). При реализации любого способа умножения предполагается, что произведение (ΣЧП) является словом двойной длины по сравнению с длиной сомножителей.

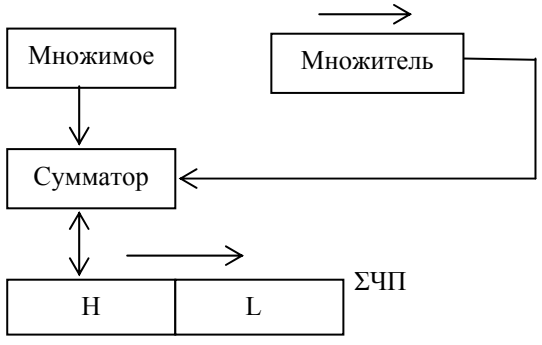
Способы умножения Таблица П6.2

	Младшими разрядами вперед		Старшими разрядами вперед	
Множитель	Сдвигается вправо	Сдвигается вправо	Сдвигается влево	Сдвигается влево
Множимое	Сдвигается влево	Не сдвигается	Сдвигается вправо	Не сдвигается
ΣЧП	Не сдвигается	Сдвигается вправо	Не сдвигается	Сдвигается влево

Специфика реализации конкретных способов умножения отображается соответствующими схемами умножения и алгоритмами их работы. При реализации умножения младшими разрядами вперед в каждом цикле сначала выполняется суммирование частичных произведений, если это необходимо, а затем сдвиг (множимого или ΣЧП). При умножении старшими разрядами вперед алгоритм изменяется. Сначала выполняется сдвиг (множимого или ΣЧП.), а затем производится суммирование частичных произведений. Независимо от используемого способа умножения в состав блока умножения должны входить регистры множителя (одинарной длины) множимого (одинарной или двойной длины), сумматора частичных произведений ΣЧП (двойной длины) и схемы реализации сдвигов.

П6.4.1. Умножение младшими разрядами вперед со сдвигом ΣЧП вправо и неподвижным множимым.

Схема умножения представлена на рис. П6.5. Особенностью данного способа умножения является то, что в нем при формировании ΣЧП используется сложение слов одинарной длины с запоминанием переноса и сдвиг ΣЧП вправо с учетом запомненного значения переноса.



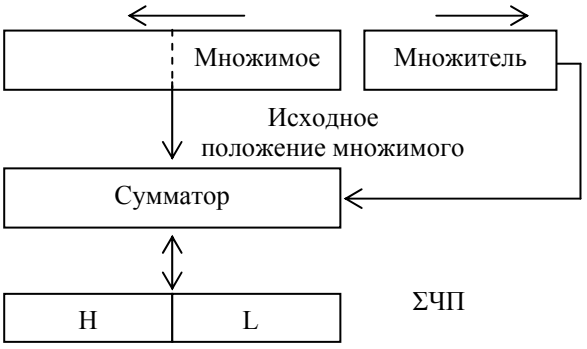
Алгоритм:

– сдвиг множителя вправо, анализ выдвигаемого бита;
– если выдвигаемый бит множителя равен 1, формирование ΣЧП с запоминанием переноса;
– сдвиг ΣЧП с учетом переноса;
– контроль числа циклов (цикл повторяется *n* раз по числу разрядов множителя)

Рис. П6.5. Умножение младшими разрядами вперед со сдвигом ΣЧП вправо и неподвижным множимым

П6.4.2. Умножение младшими разрядами вперед со сдвигом множимого влево и неподвижной ΣЧП

Схема умножения представлена на рис. П6.6. Особенностью данного способа умножения является необходимость использования регистра множимого и сумматора двойной длины по сравнению с длиной множителя.



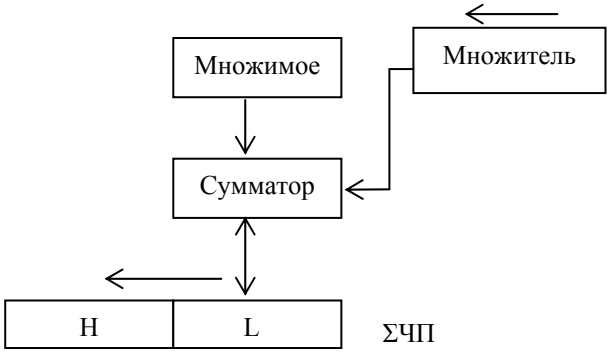
Алгоритм

– сдвиг множителя вправо, анализ выдвигаемого бита;
– если выдвигаемый бит множителя равен 1, формирование ΣЧП (суммирование двойной точности)
– контроль числа циклов (цикл повторяется *n* раз по числу разрядов множителя)

Рис. П6.6. Умножение младшими разрядами вперед со сдвигом множимого влево и неподвижной ΣЧП

П6.4.3. Умножение старшими разрядами вперед со сдвигом ΣЧП влево и неподвижным множимым.

Схема умножения показана на рис. П6.7. Особенностью данного способа умножения является необходимость учета бита переноса, формируемого при сложении младшего слова ΣЧП и множимого. Если перенос равен 1, необходимо осуществлять инкремент старшего слова ΣЧП.



Алгоритм:

– сдвиг ΣЧП влево,
– сдвиг множителя влево, анализ выдвигаемого бита;
– если выдвигаемый бит множителя равен 1, формирование ΣЧП с временным запоминанием переноса;
– контроль числа циклов (цикл повторяется *n* раз по числу разрядов множителя)

Рис. П6.7. Умножение старшими разрядами вперед со сдвигом ΣЧП влево и неподвижным множимым

Заметим, что при логических сдвигах множителя вправо или влево выход из цикла в алгоритмах П6.4.2 и П6.4.3 можно осуществить при нулевом значении множителя (при отсутствии значащих разрядов в множителе после очередного сдвига). В ряде случаев это может привести к сокращению временных затрат на реализацию умножения.

П6.4.4. Умножение старшими разрядами вперед со сдвигом множимого вправо и неподвижной ΣЧП

Схема умножения показана на рис. П6.8. Особенностью данного способа умножения является необходимость использования регистра множимого и сумматора двойной длины по сравнению с длиной множителя.

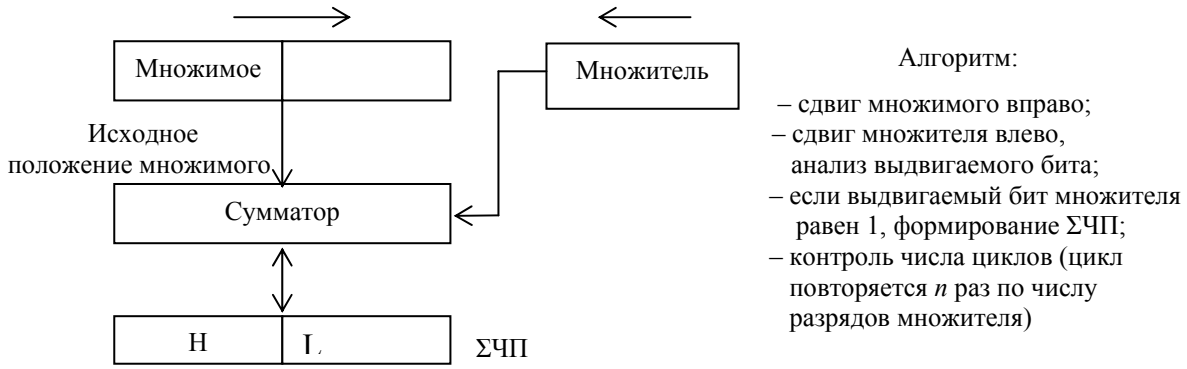


Рис. П6.8. Умножение старшими разрядами вперед
со сдвигом множимого вправо и неподвижной ΣЧП

П3.5. Деление программным способом

Деление — процесс обратный умножению. Если при умножении выполняется многократное суммирование, то при делении — многократное вычитание. Деление обычно сводится к выполнению последовательности вычитаний делителя Y сначала из делимого X, а затем из образующихся в процессе предыдущих вычитаний остатка и сдвига частичных остатков. Наиболее распространенными схемами деления являются схема деления с неподвижным делимым и сдвигаемым вправо делителем (рис. П6.9) и схема деления с неподвижным делителем и сдвигаемым влево делимым (рис. П6.10).

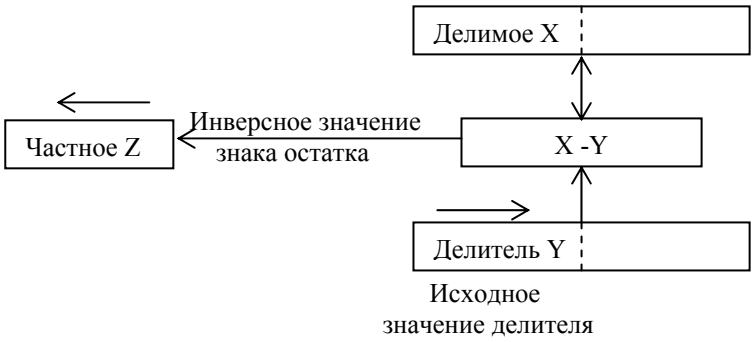


Рис. П6.9. Схема деления с неподвижным делимым и сдвигаемым вправо делителем

Вариант классического деления (с неподвижным делимым и сдвигаемым вправо делителем) на практике применяется очень редко из-за необходимости использования сумматора (вычитателя) и регистров двойной длины. В соответствии с классическим правилом деления «столбиком» частное получают, начиная со старших разрядов. Очередная цифра частного является единицей, если после вычитания делителя из остатка фиксируется положительное число, и нуль, если — отрицательное.

В схеме деления с неподвижным делителем и сдвигаемым влево делимым используется операционный блок с сумматором (вычитателем) одинарной длины и не требуются регистры двойной длины. Данный способ деления является наиболее распространенным.

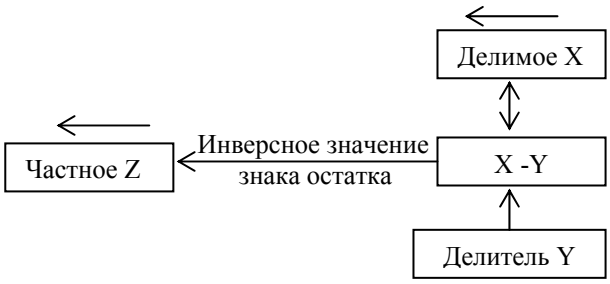


Рис. П6.10. Схема деления с неподвижным делителем и сдвигаемым влево делимым

В составе арифметико-логического блока МП содержится только сумматор (поскольку он проще вычитателя), а арифметические операции выполняются с использованием специальных кодов: прямого, обратного и дополнительного.

Реализация алгоритма деления предполагает, что исходные значения делителя и делимого нормализованы, т. е. удовлетворяют неравенству $|X| \le |Y|$. Если условие не выполняется, то предварительно (перед делением) необходимо выполнить выравнивание (масштабирование) делителя.

Процедура деления начинается с формирования модулей делимого и делителя и последующего определения и запоминания знака частного. Затем, если необходимо, осуществляется масштабирование делителя, и после этого выполняется собственно деление. Деление реализуется методом многократного (циклического) вычитания делителя Y сначала из делимого X, а затем из образующихся в процессе деления частичных остатков $X_к$ и сдвига влево частичных остатков. В микропроцессорах вычитание реализуется путем суммирования делимого с дополнительным кодом отрицательного делителя. Остаток $X_к$ может быть ≥ 0 или < 0. Если $X_к < 0$, то для формирования следующего остатка необходимо восстановить прежний остаток (путем прибавления к нему делителя в прямом коде) и перед вычитанием делителя сдвинуть его на один разряд влево. Очередная цифра частного, определяемая как инверсное значение знака частичного остатка, заносится в младший разряд частного. Число циклов вычитания делителя из частичных остатков и сдвигов делимого и частного зависит от требуемой точности выполнения операции деления (требуемой разрядности частного). Рассмотренный способ деления носит название ***деление с восстановлением остатка***. Недостатком этого способа деления при аппаратной реализации является необходимость использования дополнительного такта на восстановление остатка при $X_к < 0$.

Более быстродействующим при аппаратной реализации является ***способ деления без восстановления остатка***, не требующий восстановления предыдущего остатка после получения отрицательного остатка. При этом способе деления при выявлении отрицательного знака остатка $X_к$ сразу формируется следующий остаток $X_{к+1}$. Для этого осуществляется сдвиг остатка $X_к$ влево и его суммирование с делителем. При аппаратной реализации деление без восстановления остатка в зависимости от знака частичного остатка реализуется двумя различными циклами:

- если очередной частичный остаток $X_к > 0$, то в следующем цикле выполняется вычитание делителя из сдвинутого частичного остатка $X_к$ (путем суммирования делимого с дополнительным кодом отрицательного делителя);

- если очередной частичный остаток $X_к < 0$, то в следующем цикле выполняется суммирование сдвинутого частичного остатка $X_к$ и прямого кода делителя.

Нетрудно показать, что частичные остатки после выполнения суммирования при делении без восстановления остатка получаются такими же, как и после сдвига восстановленного остатка при делении с восстановлением остатка. Действительно, частичный остаток при делении с восстановлением остатка

$X_{к+1} = 2*(X_к + Y) - Y = 2*[(X_{к-1} - Y) + Y] - Y = 2X_{к-1} - Y$ полностью совпадает с частичным остатком при делении без восстановления остатка

$X_{к+1} = 2*(X_{к-1} - Y) + Y = 2X_{к-1} - Y.$

Следует заметить, что при реализации деления программным способом особенности рассмотренных способов деления нивелируются, и временные затраты при выполнении деления обоими способами оказываются примерно равными.

В общем случае алгоритм деления целых чисел можно описать следующей последовательностью действий:

- определяется и запоминается знак частного.

- формируются модули (прямые коды) делимого X и делителя Y. Делитель должен быть ненулевым.

- выполняется процедура нормализации (выравнивания) делителя. В зависимости от значений делителя и делимого нормализованный делитель должен удовлетворять одному из условий: $|X| < |Y|$ или $|Y| \le |X| \le 2|Y|$.

Выравнивание реализуется путем сравнения X и Y и последующих сдвигов делителя. Если в результате сравнения выясняется, что делимое X больше делителя Y (флаг переноса C=1), осуществляется сдвиг делителя влево, что соответствует его умножению на 2. Выравнивание продолжается до выполнения неравенства $|X|<|Y|$. Число сдвигов (шагов выравнивания) фиксируется. Оно определяет число разрядов целой части частного.

Если в процессе выравнивания старший (значащий) разряд делителя при очередном сдвиге вправо занимает знаковый разряд, это соответствует соотношению $|Y| \le |X| \le 2|Y|$. При таких значениях $|X|$ и $|Y|$ для правильного деления дополнительно необходимо сдвинуть делитель вправо.

- выполняется собственно деление делимого $|X|$ на нормализованный делитель $|Y|$. Первый цикл деления зависит от значений делимого и выровненного делителя. При $|X| < |Y|$ деление начинается со сдвига делимого влево, а при $|Y| \le |X| \le 2|Y|$ сдвиг делимого не выполняется. Затем из сдвинутого или несдвинутого делимого вычитается делитель. Формируемый частичный остаток $X_к$ запоминается в регистре делимого. После этого анализируется знак разности и определяется цифра, которая помещается в младший разряд частного: эта цифра равна 1, если частичный остаток $X_к \ge 0$, или 0, если $X_к < 0$. Далее контролируется число выполненных циклов, и, если деление не завершено, цикл деления повторяется, при этом перед выполнением

последующих циклов деления содержимое регистров частного и частичного остатка сдвигается влево.

- вычисленному значению частного присваивается знак, определенный ранее.

Рассмотренный алгоритм деления обеспечивает формирование модуля частного в виде правильной дроби с нулевым разрядом целой части частного. Для 8-разрядных представлений делимого и делителя 8-разрядный модуль частного формируется за 7 циклов.

В общем случае частное может содержать целую и дробную части (если выполнялась нормализация делителя). При таком представлении целая часть частного обычно размещается в старшем байте, а дробная часть – в младшем байте. По этой причине алгоритм деления должен предусматривать проверку счетчика шагов выравнивания. Если выполнялась нормализация делителя, разрядность частного должна быть увеличена. Для выделения целой части частного опрашивается счетчик шагов выравнивания, и если он не равен 0, выполняется сдвиг частного влево на число разрядов, определяемое значением этого счетчика плюс 1. Формируемое в старшем слове частного значение соответствует целой части частного.

Точность представления результата деления может быть повышена за счет увеличения числа циклов деления.

При наличии каких-либо ограничений, например использовании нормализованных значений делимого и делителя, алгоритм деления упрощается.

П6.6. Транспонирование матрицы, содержащей n строк и n столбцов

Матрица представляет собой двумерный массив данных, в котором доступ к любому из элементов осуществляется по двум индексам — номеру столбца i и номеру строки j. Адрес элемента матрицы будем обозначать a_{ji} при отображении по строкам и – при отображении по столбцам. В памяти элементы матрицы хранятся в виде одномерного массива данных с линейной организацией. Матрица может размещаться в произвольной области памяти. Для определенности будем считать, что она размещается в младших адресах 256-байтной страницы, т. е. адрес ее первого элемента в странице равен нулю. Для получения адреса произвольного элемента матрицы его необходимо преобразовать в линейный адрес. При принятых ограничениях линейный адрес элемента строки определяется по формуле $a_{ji} = i + n*j$, а адрес элемента столбца — по формуле $a_{ij} = j + n*i$, где n — число элементов в строке (столбце), $0 \le i, j \le n$.

Общим методом решения задачи транспонирования является перестановка элементов строк a_{ji} и столбцов a_{ij} , в результате которой строки и столбцы меняются местами.

Для практической реализации перестановок необходимо вычислить линейные адреса переставляемых элементов строк и столбцов и осуществить обмен содержимым ячеек памяти по вычисленным адресам. После обмена элементов первой строки и первого столбца исходной матрицы n-го порядка

виртуально легко выделяется матрица (n – 1)-го порядка, в которой элементы первой строки аналогично заменяются элементами первого столбца. В результате выделяется матрица (n – 2)-го порядка. Преобразование выполняется до выделения матрицы 0-го порядка.

Описанный алгоритм реализуется в виде двух вложенных циклов: в первом цикле осуществляются перестановки элементов строки и столбца (инкремент i). Этот цикл является вложенным. Во втором (внешнем) цикле выполняется определение порядка преобразуемой матрицы (инкремент j).

П6.7. Определение n-го члена и суммы n членов ряда Фибоначчи

Числа Фибоначчи — элементы числовой последовательности, в которой первые два члена равны 1, а каждый последующий член равен сумме двух предыдущих. В соответствии с этим определением значение (n + 2)-го члена a_{n+2}, начиная с 3-го члена, вычисляется по формуле $a_{n+2} = a_{n+1} + a_n$, где n = 1, 2, ..., $a_1 = a_2 = 1$

Для вычисления значений членов ряда Фибоначчи целесообразно использовать циклическую процедуру.

При инициализация цикла осуществляется:

– загрузка значения m в счетчик числа итераций цикла (нетрудно заметить, что для (n + 2)-го члена значение m должно быть на 2 меньше искомого номера члена ряда);

– загрузка известных по определению значений двух первых членов ряда в ячейки памяти, зарезервированные для (n + 1)-го и n-го членов ряда.

Собственно выполнение цикла включает:

– вычисление значения a_{n+2} (n + 2)-го члена ряда;

– декремент счетчика итераций и проверка окончания цикла. Если 0, то завершение процедуры вычисления искомого члена ряда и запись результата. В противном случае, обновление исходных данных для следующей итерации. С этой целью ранее вычисленное значение (n + 1)-го члена ряда помещается в ячейку для n-го члена ряда, а на место (n + 1)-го члена помещается вычисленное значение a_{n+2} (n + 2)-го члена ряда. После этого осуществляется переход к началу выполнения цикла.

При вычислении суммы значений S_n n членов ряда Фибоначчи также можно воспользоваться циклической процедурой, в каждом цикле которой, наряду с вычислением значения a_{n+2} (n + 2)-го члена ряда, дополнительно осуществляется прибавление вычисленного значения a_{n+2} к сумме ранее вычисленных значений.

Более простым способом вычисления суммы S_n может оказаться вычисление S_n по эмпирической формуле

$$S_n = a_{n+2} - 1, \quad \text{где} \quad n = 1, 2, \ldots$$

П6.8.Определение делимости на три 8-разрядного двоичного числа

В общем случае признак делимости на три зависит от используемой системы счисления. Для двоичной системы счисления правило делимости на три формулируется следующим образом. Двоичное число делится на три, если

сумма цифр, записанных в четные разряды, равна сумме цифр, записанных в нечетные разряды, либо отличается на число, делящиеся на три. Для восьмиразрядных беззнаковых целых чисел разность указанных сумм не может быть больше четырех, при этом среди возможных вариантов только два соответствуют делимости байтового числа на три.

П6.9. Определение числа значащих (незначащих) разрядов в двоичном коде числа

Для решения задачи можно использовать циклическую процедуру, в каждом цикле которой необходимо осуществлять проверку значения одного разряда кода и модифицировать результат в соответствии с этой проверкой. Последовательный опрос разрядов легко реализовать путем сдвига двоичного числа и анализа выдвигаемого бита. Требуемое количество циклов зависит от разрядности двоичного кода числа, и для его хранения необходимо предусмотреть специальный счетчик циклов. В процессе выполнения программы значение счетчика циклов должно изменяться после каждого цикла и анализироваться для определения окончания процедуры вычисления.

П6.10. Определение числа совпадающих (несовпадающих) разрядов двух n-разрядных двоичных чисел

При решении задачи необходимо выполнить сравнение одноименных разрядов двух двоичных чисел. Сравнение удобно выполнить с помощью поразрядной операции ИСКЛЮЧАЮЩЕЕ ИЛИ (сложение по модулю 2). Результатом этой операции является двоичное число, разряды которого содержат 1 для несовпадающих разрядов исходных чисел и 0 — для совпадающих. Нетрудно заметить, что после преобразований задача сводится к задаче, рассмотренной в п. П3.9.

П6.11. Определение числа переходов из 0 в 1 и из 1 в 0 в двоичном коде числа.

Решение данной задачи может использоваться при обработке цифровых данных экспериментальных исследований для определения числа периодов обрабатываемого сигнала. Фактически решается задача определения числа фронтов и спадов сигнала.

Сравнительно простым алгоритмом решения задачи определения числа переходов в слове, начиная со старшего разряда, является следующий:

– исходное слово сдвигается арифметически вправо на один разряд;

– реализуется сложение по mod 2 значения сдвинутого числа с исходным. Нетрудно заметить, что количество единиц в слове после выполнения указанной операции получается равным числу переходов из 0 в 1 и из 1 в 0 в двоичном коде исходного числа.

– выполняется подсчет числа единиц в слове, например с использованием решения, предложенного в п. П6.9.

При определении числа переходов в слове, начиная с младшего разряда, алгоритм оказывается сложнее, поскольку в системах команд микропроцессоров отсутствует команда (микрокоманда) «размножения»

младшего разряда числа при сдвиге влево, аналогичная команде арифметического сдвига вправо. При выполнении сдвига влево указанную операцию сохранения значения младшего разряда числа можно реализовать некоторой последовательностью команд. После чего реализуется сложение по mod 2 значения сдвинутого числа с исходным и выполняется подсчет числа единиц в результирующем слове.

П6.12. Определение числа переходов из 0 в 1 (или из 1 в 0) в двоичном коде числа.

Существует несколько вариантов решения.

В качестве первых шагов решения данной задачи можно выделить все переходы из 0 в1 и из 1 в 0 в исходном слове (см. п. П6.11). Для этого требуется выполнить сложение по mod 2 двух слов — исходного и сдвинутого арифметически на один разряд вправо. Единицы в разрядах результата этой операции соответствуют всем переходам в исходном слове. Для выделения переходов, определяемых условием задачи, необходимо на слово результата операции сложения по mod 2 наложить маску. При выполнении операции И с исходным словом отмечаются переходы из 1 в 0, а при выполнении операции И с инверсным значением исходного слова выделяются переходы из 0 в 1. Число единиц в слове результата маскирования соответствует числу искомых переходов.

Более простым в реализации может оказаться алгоритм циклической процедуры, реализующий следующую последовательность действий:

– на исходное слово накладывается маска, с помощью которой выделяются два старших разряда, а остальные разряды слова обнуляются;

– результат сравнивается с заданным типом перехода (10 0…0 или 01 0…0). При равенстве значений выделенных разрядов с заданным типом фиксируется переход, в противном случае переход не фиксируется;

– анализируемое слово сдвигается влево на один разряд, модифицируется счетчик циклов, после чего выполняется проверка условия завершения цикла. При невыполнении условия цикл повторяется для нового значения исходного слова.

Еще один алгоритм циклической процедуры для определения числа переходов заданного типа предусматривает выделение и анализ значения старшего разряда исходного слова. Если это значение соответствует первой цифре типа перехода, то осуществляется переход к ветви, в которой выделяется следующий разряд исходного слова и анализируется его значение на равенство второй цифре типа перехода. С этой целью исходное слово сдвигается влево на один разряд и определяется значение старшего разряда слова после сдвига. В зависимости от его значения фиксируется или не фиксируется переход. Если значение старшего разряда исходного слова не соответствует первой цифре типа перехода, то исходное слово просто сдвигается влево на один разряд. Независимо от выбираемой ветви при завершении цикла модифицируется счетчик циклов и выполняется проверка условия завершения цикла. При невыполнении условия цикл повторяется для нового значения исходного слова.

П6.13. Определение максимальной последовательбности «1» («0») в двоичном коде числа

Для определения максимальной последовательбности единиц (нулей) в двоичном коде числа необходимо использовать два счетчика — счетчик текущей последовательности и выходной результирующий счетчик. Задачу можно решить способом, рассмотренным в п. П3.10 с помощью циклической процедуры из n циклов (n — разрядность кода анализируемого числа). Исходное число (при подсчете «1») или инвертированное число (при подсчете «0») сдвигается влево на один разряд, и анализируется выдвигаемый бит. При единичном значении этого бита осуществляется инкремент счетчика текущей последовательности $N_{тек}$ и сравнение значения этого счетчика со значением выходного счетчика $N_{вых}$. Если $N_{тек} > N_{вых}$, значение счетчика текущей последовательности переписывается в выходной счетчик, в противном случае значение $N_{вых}$ остается неизменным. При любом значении выдвигаемого бита (независимо от выбираемой ветви) при завершении цикла модифицируется счетчик циклов и выполняется проверка условия завершения цикла. При невыполнении условия цикл повторяется.

Порядок подготовки и проведения лабораторных НИР

Лабораторные НИР по курсу с обобщенным названием «Организация ЭВМ, микропроцессоры и микропроцессорные системы» помогают закрепить теоретические знания соответствующих учебных дисциплин. Выполнение НИР обеспечивает практическое освоение методов построения микропроцессорных систем, способствует приобретению навыков решения прикладных задач с использованием микропроцессоров разных типов, знакомит с программированием в машинных кодах.

Отдельные исследования по темам лабораторного практикума выполняются студентами индивидуально в течение четырехчасовых посещений в лаборатории.

Подготовка к проведению НИР. Выполнению НИР должна предшествовать обязательная домашняя подготовка. В процессе подготовки к НИР по материалам разделов 1–3 данного пособия изучаются принципы организации и особенности функционирования исследуемого микропроцессора. Итогом домашней подготовки является макет отчета по НИР, содержащий структурные схемы и краткое описание работы МП и исследуемой микропроцессорной системы, алгоритмы и программы (микропрограммы) тестовых и индивидуальных заданий, протокол исследований с таблицами для записи результатов. Наличие макета отчета является основанием для допуска к выполнению исследований в лаборатории.

Проведение НИР. Цикл работ по конкретному микропроцессору предваряет контрольная работа, являющаяся своеобразным индикатором готовности студента к проведению исследований. Положительный результат тестирования является одной из составляющих зачета по теме исследования. Варианты контрольных работа приводятся ниже.

При выполнении НИР результаты исследований и данные по отладке программ заносятся в протокол отчета. Записи в отчете должны быть четкими и ясными. Выполняемые программы должны быть снабжены поясняющими комментариями.

Важной составляющей исследований является освоение методики поиска и локализации неисправностей, возникающих при работе микропроцессорной системы (стенда). Особенности функционирования конкретного стенда, выявленные в процессе работы, должны быть отражены в отчете.

Исследования конкретного микропроцессора предполагают их выполнение в течение 3-х — 4-х лабораторных занятий. Работа считается выполненной после подписи преподавателем протокола отчета. Единый (окончательно оформленный) отчет по законченному исследованию конкретного микропроцессора представляется к проверке и защите не позднее начала работ по исследованию другого МП.

Контрольное задание (вариант) по изучению МП 8080А.

1. Какими командами можно выполнить следующие действия (укажите восьмеричный код команды и ее мнемонику)?

1.1. Загрузка регистровой пары DE в стек;

1.2. Сложение по mod 2 содержимого аккумулятора с числом 16 h.

1.3. Условный переход по нулевому результату по адресу 1009 h.

1.4. Программно запретить прерывания.

2. По мнемоническому описанию команд STC; SUB D; ANI B2; STA addr; ADD M; MOV M,C; INX H указать словесное описание команды; число байт в команде; число машинных циклов выборки и исполнения команды (в соответствии с табл.1); способы адресации каждого из операндов и результата (в соответствии с табл.2).

Число машинных циклов выборки и исполнения команды Таблица 1

Мнемоника команды	Количество машинных циклов		
	выборка	исполнение	всего
OUT port	2	1	3

Способы адресации операндов Таблица 2

Мнемоника команды	Выполняемая операция	Способ адресации		
		Операнд 1	Операнд 2	Приемник
STC	1 → C	неявная	-	неявная

3. Содержимое H = 007_o, L = 377_o.

Какое значение будет зафиксировано в регистрах H и L после выполнения команды INR H?

4. Определите содержимое слова состояния PSW (регистра признаков и аккумулятора) после выполнения каждой команды программы. Ответ представьте в виде таблицы:

Мнемоника команды	Аккумулятор	Флаги S Z 0 C` 0 P 1 C
mvi B 200		
mvi A 001		
xra B		
rlc		
lxi SP 300 000		
push PSW		
hlt		

В каких ячейках памяти будут сохранены значения аккумулятора и регистра флагов после выполнения программы?

Контрольное задание (вариант) по изучению модульного МП К1804.

1. Для заданных условно-мнемоническим описанием микрокоманд заполните операционные поля для секции ЦПЭ К1804ВС1 в виде таблицы с нижеприведенным форматом.

РОН9 := (РОН9) - (РОН1)
PQ := 13d
PQ := Dh ⊕ (Q)
$ШД_{вых}$:= (РОН8)
РОН4 := 2(РОН4)
РОН2 := R1(РОН1)

Приемник результата	Источник операндов	C0	**Операция АЛУ**	A	B	D

2. Заполните операционные поля для секции ЦПЭ К1804ВС1 в виде таблицы с нижеприведенным форматом для выполнения следующих действий:
   2.1. Логически сложить содержимое РОН3 и РОН7.
   2.2. Записать в РОН0 число 1101b.
   2.3. Обнулить содержимое РОН5.

Приемник результата	Источник операндов	C0	**Операция АЛУ**	A	B	D

3. Как изменится формат операционного поля микрокоманды для 8-разрядного микропроцессора на базе К1804 (две секции ЦПЭ К1804ВС1)?

4. Укажите коды инструкций переходов:
   4.1. Возврат из подпрограммы.
   4.2. Остановка программы (с комментарием-пояснением).
   4.3. Переход на адрес из регистра микрокоманд по знаку числа.

Рекомендуемое содержание отчета.

Отчет должен содержать титульный лист и собственно отчет.

На титульном листе необходимо указать тему НИР, дату ее проведения, фамилию, имя, отчество студента и номер его академической группы. (Образец представлен ниже).

Собственно отчет должен включать следующие разделы:

– цель исследования;

– программу исследований;

– структурные схемы и краткое описание работы МП и исследуемой микропроцессорной системы;

– программы (микропрограммы) работы тестовых и индивидуальных заданий;

– протокол исследований с таблицами результатов и комментариями по каждому пункту программы исследований;

– анализ полученных результатов на каждом этапе исследования и общие выводы по результатам работы в целом.

Образец титульного листа

Санкт-Петербургский государственный политехнический университет

Институт информационных технологий и управления

Кафедра компьютерных систем и программных технологий

Отчет по лабораторной работе

(Учебная дисциплина «Вычислительные машины, системы и сети» для группы 33501/1 или «ЭВМ и периферийные устройства» для групп 33501/3,4)

Исследование однокристального микропроцессора с фиксированным списком команд

или

Исследование секционного микропроцессора с микропрограммным управлением, с естественной, принудительной и стековой адресацией микрокоманд

Работу выполнил студент группы №_____ ФИО_____

Работу принял преподаватель _____ ФИО_____

Санкт-Петербург

201_

ПАВЛОВСКИЙ Евгений Григорьевич
ЖВАРИКОВ Владимир Анатольевич
КУЗЬМИН Александр Александрович

ОРГАНИЗАЦИИ И ОСНОВЫ ПРОГРАММИРОВАНИЯ МИКРОПРОЦЕССОРОВ

Учебное пособие и методические указания к лабораторному практикуму