ВВЕДЕНИЕ

Актуальность. В настоящее время в системах управления и обработки данных все чаще применяются микроконтроллеры, решающие широкий спектр задач. Однокристальные микроконтроллеры (ОМК) являются наиболее массовым видом устройств современной микропроцессорной техники. Интегрируя на одном кристалле высокопроизводительный процессор, память и набор периферийных схем, ОМК позволяют с минимальными затратами реализовать высокоэффективные системы и устройства управления различными объектами (процессами). В отличие от обычных микропроцессоров, для работы которых необходимы внешние интерфейсные схемы, в корпусе ОМК наряду с основными функциональными узлами размещены такие вспомогательные узлы, как тактовый генератор, таймер, контроллер прерываний, цифро-аналоговый и аналого-цифровой преобразователи, порты ввода-вывода.

Система управления, в зависимости от сложности решаемых ею задач, может быть реализована на основе специализированной интегральной схемы, многокристальной микропроцессорной системы (МПС) или на основе ОМК. Применение МПС, выполненной в виде модульной конструкции, или однокристального микроконтроллера позволяет «встроить» систему управления непосредственно в управляемый объект. Поэтому для характеристики систем управления данного класса в настоящее время широко используется термин «встроенная система». Этот термин обозначает вычислительную систему, входящую неотъемлемой частью в состав другой системы, формирующий управляющий компонент какой-либо технической метасистемы, который должен функционировать при минимальном вмешательстве человека.

Основными характеристиками встраиваемых систем являются:

- ограниченность аппаратных ресурсов;
- неизменность программного обеспечения в процессе применения;
- высокая надежность;
- короткий цикл разработки и внедрения;
- серийность производства;
- низкая себестоимость.

Именно благодаря этим качествам ОМК находят широкое применение в системах промышленной автоматики, контрольно-измерительных приборах и системах, аппаратуре связи, автомобильной электронике, медицинском оборудовании, бытовой технике и многих других областях.

Применение однокристальных микроконтроллеров позволяет перенести основные затраты, связанные с разработкой встраиваемых систем управления, из аппаратной в программную область, что влечет за собой увеличение сложности программного обеспечения (ПО) микроконтроллеров.

При проектировании микроконтроллеров приходится соблюдать баланс между размерами и стоимостью с одной стороны и гибкостью и производительностью с другой. Для разных приложений оптимальное соотношение этих и других параметров может различаться очень сильно. Поэтому существует огром-

ное количество типов микроконтроллеров, отличающихся архитектурой процессорного модуля, размером и типом встроенной памяти, набором периферийных устройств, типом корпуса и т. д.

В современных ОМК применяются следующие архитектуры процессоров:

- RISC — (Reduce Instruction Set Commands) архитектура с сокращенным набором команд;
- CISC — (Complex Instruction Set Commands) традиционная архитектура с расширенным набором команд;
- ARM — (Advanced RISC — machine) усовершенствованная RISC архитектура.

Большое распространение получили микроконтроллеры с RISC-архитектурой (англ. Reduced Instruction Set Computer — вычисления с сокращённым набором команд). Сокращенный набор команд позволяет выполнять большинство инструкций за один такт, что обеспечивает высокое быстродействие даже при относительно низкой тактовой частоте.

Главная задача RISC архитектуры обеспечение наивысшей производительности процессора. Её отличительными чертами является:

- малое число команд процессора (несколько десятков);
- каждая команда выполняется за минимальное время (1-2 машинных цикла, такта).
- максимально возможное число регистров общего назначения процессора (несколько тысяч);
- увеличенная разрядность процессора (12,14,16 бит).

Современная RISC архитектура включает, как правило, только последние 3 пункта, так как за счет повышенной плотности компоновки БИС стало возможным реализовать большое количество команд.

В современных 32-разрядных ОМК используют ARM архитектуру (расширенная RISC архитектура с суперсокращением команд THUMB).

В то время как 8-разрядные процессоры общего назначения полностью вытеснены более производительными моделями, 8-разрядные микроконтроллеры продолжают широко использоваться. Это объясняется тем, что существует большое количество применений, в которых не требуется высокая производительность, но важна низкая стоимость. В то же время, есть микроконтроллеры, обладающие большими вычислительными возможностями, например цифровые сигнальные процессоры.

Ограничения по цене и энергопотреблению сдерживают также рост тактовой частоты контроллеров. Хотя производители стремятся обеспечить работу своих изделий на высоких частотах, они, в то же время, предоставляют заказчикам выбор, выпуская модификации, рассчитанные на разные частоты и напряжения питания. Во многих моделях микроконтроллеров используется статическая память для ОЗУ и внутренних регистров. Это даёт контроллеру возможность работать на меньших частотах и даже не терять данные при полной остановке тактового генератора. Часто предусмотрены различные режимы

энергосбережения, в которых отключается часть периферийных устройств и вычислительный модуль.

Неполный список периферии, которая может присутствовать в микроконтроллерах, включает в себя:

- универсальные цифровые порты, которые можно настраивать как на ввод, так и на вывод;
- различные интерфейсы ввода-вывода, такие как UART, I²C, SPI, CAN, USB, IEEE 1394, Ethernet;
- аналого-цифровые и цифро-аналоговые преобразователи;
- компараторы;
- широтно-импульсные модуляторы;
- таймеры;
- контроллеры бесколлекторных двигателей;
- контроллеры дисплеев и клавиатур;
- радиочастотные приемники и передатчики;
- массивы встроенной флэш-памяти;
- встроенный тактовый генератор и сторожевой таймер.

Программирование микроконтроллеров обычно осуществляется на языке ассемблера или Си, хотя существуют компиляторы для других языков, например, Форта. Используются также встроенные интерпретаторы Бейсика. Для отладки программ используются программные симуляторы (специальные программы для персональных компьютеров, имитирующие работу микроконтроллера).

1. Функциональное назначение выводов корпуса МК48

Микроконтроллер конструктивно выполнен в корпусе БИС с 40 внешними выводами. Все выводы электрически совместимы с элементами ТТЛ: входы представляют собой единичную нагрузку, а выводы могут быть нагружены одной ТТЛ-нагрузкой. Цоколевка корпуса МК48 показана на рисунке 1. Ниже приводятся символические имена выводов корпуса и даются краткие пояснения их назначения. Звездочкой будут помечены инверсивные входы/выходы.

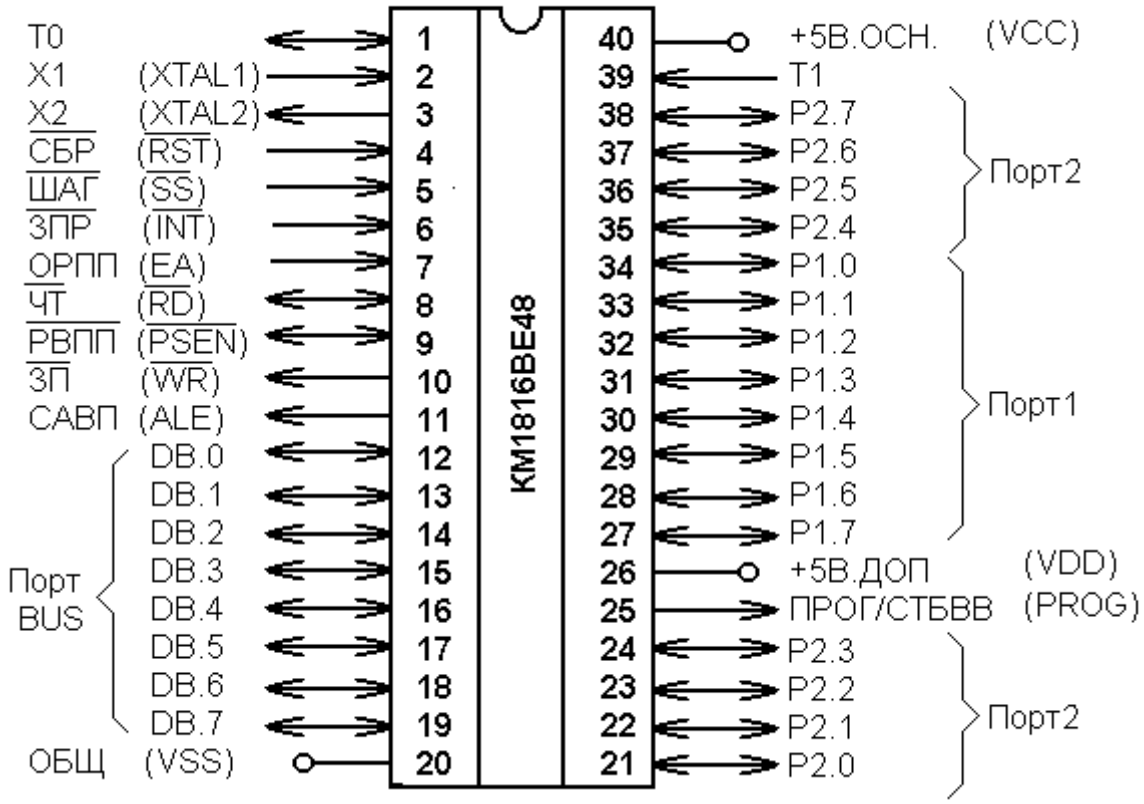

Рис. 1 –Цоколевка корпуса МК48

ОБЩ (VSS) потенциал земли;

+5.ОСН (VCC) основное напряжение питания +5 В, подается во время работы и при программирование РПП;

+5.ДОП (VDD) дополнительное напряжение питания +5 В, во время работы обеспечивает питание только для РПП, на этот вывод при программирование попадает питание +25 В;

ПРОГР (PROG) вход для подачи программирующего импульса +25 В при загрузке РПП, вход стробирующего сигнала для БИС расширителя ввода/вывода;

Х1 вход для подключения вывода кварцевого резонатора или вход для сигнала от внешнего источника синхронизации;

Х2 вход для подключения второго вывода резонатора;

***СБР (RST)** вход сигнала общего сброса при запуске МК, сигнал 0 при программирование и проверке РПП;

***ШАГ (SS)** сигнал, который совместно с сигналом САВП позволяет при отладке выполнять программу с остановом после исполнения очередной команды;

***РВПП (PSEN)** разрешение внешней памяти программ, сигнал выдается только при обращении к внешней памяти программ;

САВП (ALE) строб адреса внешней памяти, сигнал используется для приема и фиксации адреса внешней памяти на внешнем регистре, сигнал является идентификатором машинного цикла, так как всегда выводится из МК с частотой в 5 раз меньшей основной частоты синхронизации;

***ЧТ (RD)** стробирующий сигнал при чтении из внешней памяти данных;

***ЗП (WR)** стробирующий сигнал при записи во внешнюю память данных;

Т0 входной сигнал опрашиваемый по командам условного перехода JT0 и JNT0; кроме того используется при программировании РПП, может бать использован для вывода сигнала синхронизации после команды ENT0 CLK;

Т1 входной сигнал, опрашиваемый командами условного перехода JT1 и JNT1, кроме того, используется в качестве входа внутреннего счетчика внешних событий после исполнения команды STRT CNT;

***ЗПР (INT)** сигналы запроса прерывания от внешнего источника, вызывает подпрограмму обслуживания прерывания, если прерывание разрешено ранее по команде EN I, сигнал *СБР запрещает прерывания;

ОРПП (EA) отключение РПП, уровень 1 заставляет МК выполнять выборку команд только из внешней памяти программ, используется при тестировании прикладных программ и отладки МК, подается 25 В при программировании РПП;

Порт 1 (P1) 8-битный квазидвунаправленый порт ввода/вывода информации, каждый разряд порта может быть запрограммирован на ввод или вывод;

Порт 2 (P2) 8-битный квазидвунаправленый порт ввода/вывода информации, каждый разряд порта может быть запрограммирован на ввод или вывод, биты 0-3 этого порта вовремя чтения из ВПП содержат старшие четыре бита счетчика команд СК 8-11, используется для подключения БИС расширителя ввода/вывода (порты Р4 - Р7);

Порт BUS (DB) 8-битный двунаправленый порт ввода/вывода информации, может быть отключен от нагрузки, может выполнять прием и выдачу байтов синхронно с сигналами *ЧТ и *ЗП, при обращении к ВПП содержит 8 младших бит счетчика команд и затем по сигналу *РВПП принимает выбранную команду, при обращении к ВПД содержит младшие 8 бит адреса синхронно с сигналом САВП и байт данных синхронно с сигналами *ЧТ или *ЗП.

2. Структурная схема МК48

На рисунке 2 показана структурная схема МК48. Основу структуры МК образует внутренняя двунаправленная 8-битная шина, которая связывает между собой все устройства МК:

- арифметико-логическое устройство (АЛУ);
- устройство управления;
- память и порты ввода/вывода информации.

Рассмотрим последовательно основные элементы структуры и особенности организации МК.

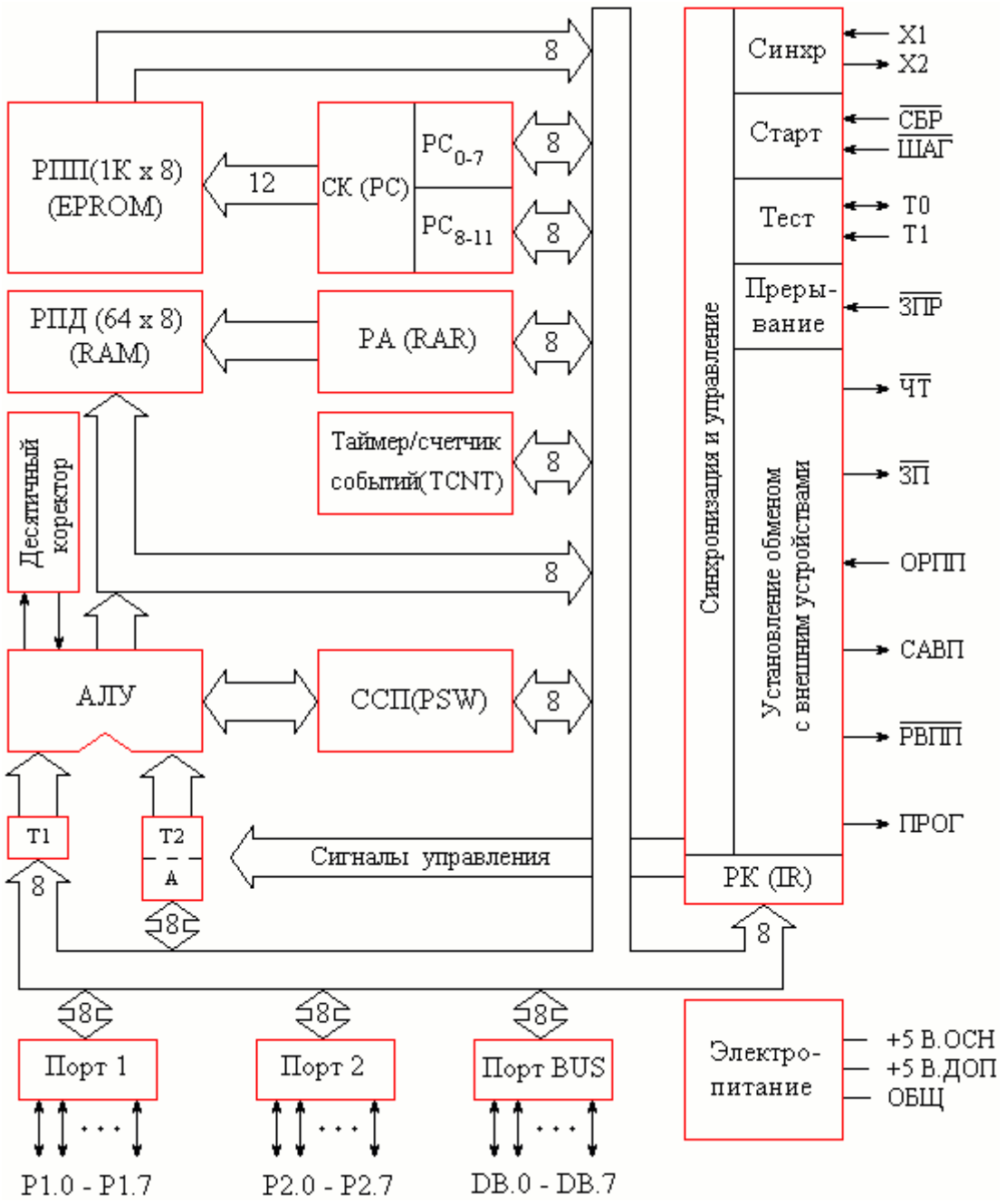

Рис. 2. Структурная схема МК48

2.1. Арифметико-логическое устройство

В состав АЛУ входят следующие блоки: комбинационная схема обработки байтов, регистры T, регистр-аккумулятор A, схема десятичного корректора и схема формирования признаков. Аккумулятор используется в качестве регистра операнда и регистра результата. Регистр временного хранения операнда T1 программно недоступен и используется для временного хранения второго операнда при выполнении двухоперандных команд. Комбинационная схема АЛУ может выполнять следующие операции: сложение байтов с переносом или без него; логические операции И, ИЛИ и исключающее ИЛИ; инкремент, декремент, инверсию, циклический сдвиг влево, вправо через (или минуя) флаг переноса, обмен тетрад в байте; десятичную коррекцию содержимого аккумулятора.

При выполнении операций обработки данных в АЛУ вырабатываются флаги (признаки), которые (за исключением флага переноса C) формируются на комбинационной схеме и не фиксируются на триггерах. К таким флагам относятся флаг нулевого содержимого аккумулятора и флаг наличия единицы в селектируемом бите аккумулятора. Логика условных переходов по указанным флагам позволяет выполнять команды передачи управления (JZ, JNZ, JB0 - JB7) без их фиксации на триггерах.

Флаги переноса и вспомогательного переноса (перенос из младшей тетрады в старшую) фиксируются на триггерах, входящих в состав регистра слова состояния программы (ССП). Формат ССП показан на рисунке 3. Кроме перечисленных признаков логика условных переходов МК оперирует флагами FO и FI, функциональное назначение которых определяется разработчиком: флагом переполнения таймера TF, сигналами на входах T0 и T1. Программистом могут быть также использованы флаги рабочего банка регистров BS и выбранного банка внешней памяти программ MB. Кроме того, логикой переходов после окончания каждого машинного цикла опрашивается еще один флаг, а именно флаг разрешения/запрета прерывания.

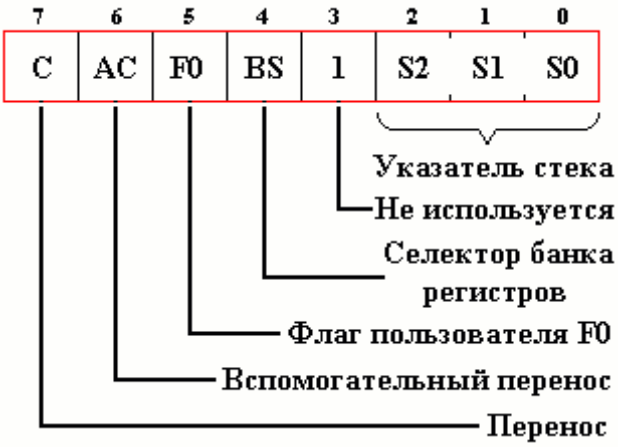

Рис. 3 Формат слова состояния программы (ССП)

2.2. Память микроконтроллера

Память программ. Память программ и память данных в МК48 физически и логически разделены. Память программ реализована в резидентном СППЗУ емкостью 1 Кбайт. Максимальное адресное пространство, отводимое для программ, составляет 4 Кбайта. Счетчик команд (СЧ) содержит 12 бит, но инкрементируются в процессе счета только младшие 11 бит. Поэтому счетчик команд из предельного состояния 7FFH (если только по этому адресу не расположена команда передачи управления) перейдет в состояние 000H. Состояние старшего бита счетчика команд может быть изменено специальными командами (SEL MB0, SEL MB1). Подобный режим работы счетчика команд позволяет создать два банка памяти емкостью по 2 Кбайта каждый. Карта адресов памяти программ показана на рисунке 4.

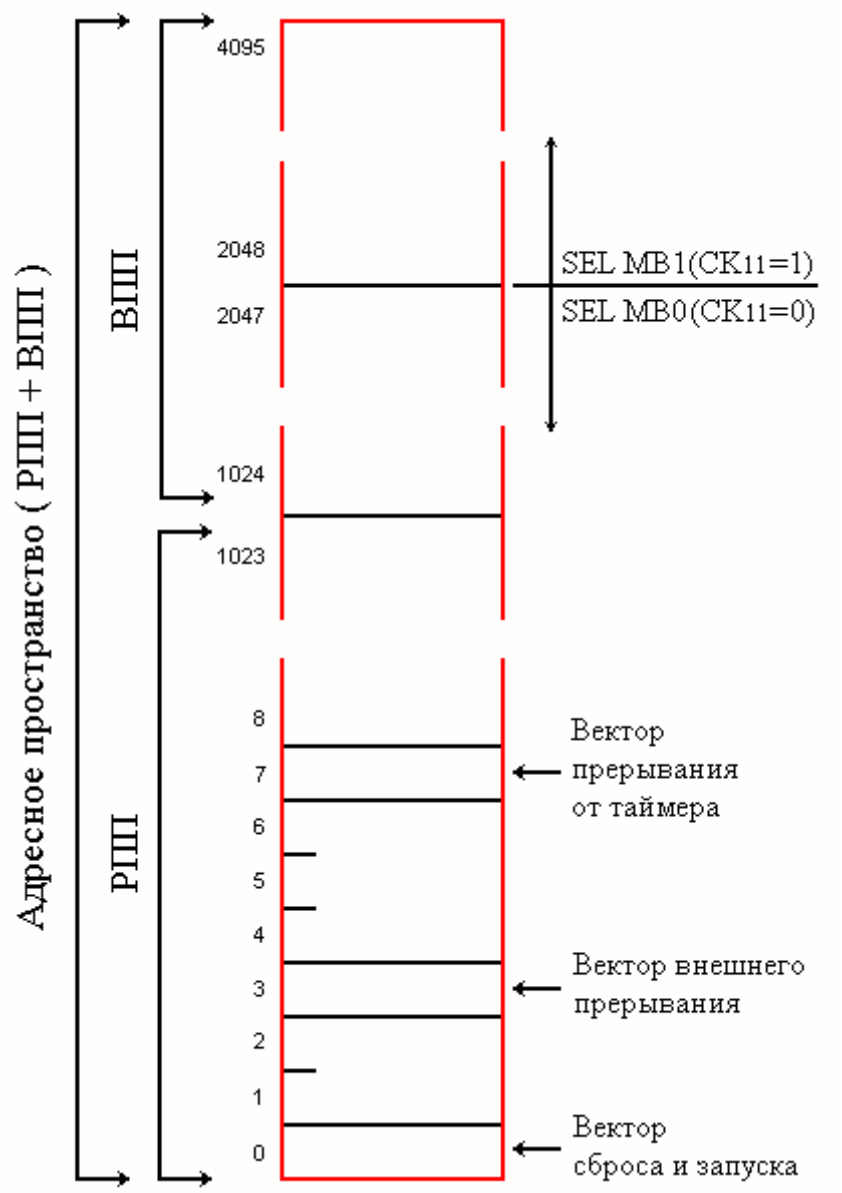

Рис. 4. Карта адресов памяти программ

В резидентной памяти программ имеется три специализированных адреса:

- адрес 0, к которому передается управление сразу после окончания сигнала СБР, по этому адресу должна находиться команда безусловного перехода к началу программы;
- адрес 3, по которому расположен вектор прерывания от внешнего источника;
- адрес 7, по которому расположены вектор прерывания от таймера или начальная команда подпрограммы обслуживания прерывания по признаку переполнения таймера-счетчика.

Память программ разделяется не только на банки емкостью 2 Кбайта, но и на страницы по 256 байт в каждой. В командах условного перехода задается 8-битный адрес передачи управления в пределах текущей страницы. В случае, когда в программе необходимо иметь много переходов по условию, из-за небольшого размера страницы возникает проблема размещении соответствующих программных модулей в границах страницы. Команда вызова подпрограмм модифицирует 11 бит счетчика команд, обеспечивая тем самым межстраничные переходы в пределах выбранного банка памяти программа.

В МК-системе, работающем с внешней памятью программ, возникает проблема размещения подпрограмм в двух банках памяти. Проблема эта связана с тем, что МК не имеет средств считывания и анализа флага МВ, равного содержимому старшего бита счетчика команд СК11. По этому, в каждый текущий момент исполнения программы, состоящей из потока вызовов подпрограмм, нет возможности определения номера банка памяти, из которого осуществляется выборка. Так как переходы между банками выполняются только по командам SEL MB, необходимо следить за тем, чтобы подпрограммы, взаимно вызывающие друг друга, располагались в пределах одного банка памяти. В противном случае возникает необходимость модификации признака МВ в вызываемой подпрограмме и восстановления его при возврате в вызывавшую подпрограмму. Но если вызов такой подпрограммы носит условный характер, то проблема восстановления может оказаться неразрешимой.

При обработке запросов прерываний в МК48 старший бит счетчика команд СК11 принудительно устанавливается в 0. Это приводит к необходимости подпрограмму обслуживания прерывания и все подпрограммы, вызываемые ею, размещать в пределах банка памяти 0.

Память данных. Резидентная память данных емкостью 64 байта имеет в своем составе два банка рабочих регистров 0-7 и 24-31 по восемь регистров в каждом. Выбор одного из банков регистров выполняется по команде SEL RB. Рабочие регистры доступны по командам с прямой адресацией, а все ячейки РПД доступны по командам с косвенной адресацией. В качестве регистров косвенного адреса используются регистры R0, R1 и R0*, R1* (рис. 5).

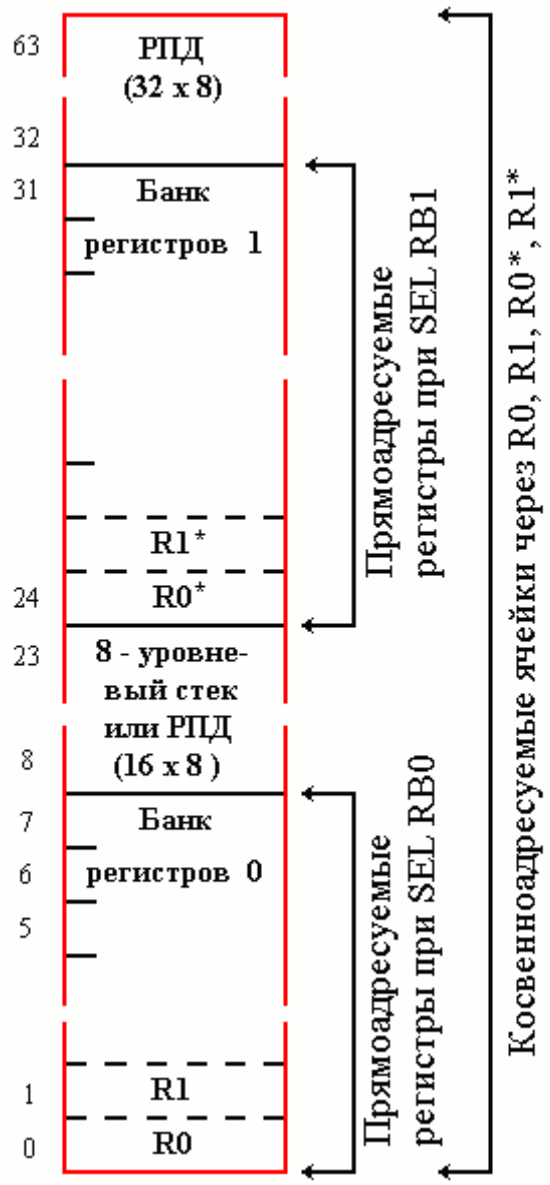

Рис. 5. Карта адресов памяти данных (РПД)

Ячейки РПД с адресами 8-23 адресуются указателем стека из ССП и могут быть использованы в качестве 8-уровневого стека. В случае если уровень вложенности подпрограмм меньше восьми, незадействованные в стеке ячейки могут использоваться как ячейки РНД. При переполнении стека, регистр-указатель стека, построенный на основе 3- битного счетчика, переходит из состояния 7 в состояние 0. Малая емкость стека ограничивает число возможных внешних источников прерывания в МК-системе. МК48 не имеет команд загрузки байта в стек или его извлечения, и в нем фиксируется только содержимое счетчика команд и старшая тетрада ССП (флаги). В силу этого разработчику необходимо следить за тем, чтобы вложенные подпрограммы не использовали одни и те же рабочие регистры.

Практически все команды с обращением к РПД оперируют с одним байтом. Однако по командам вызова и возврата осуществляется доступ к двухбайтным словам. В памяти данных слова хранятся так, что старший байт слова располагается в ячейке с большим адресом. Отметим, что в памяти программ порядок

расположения байтов по старшинству при хранении двухбайтных слов обратный.

В МК-системах, где используется внешнее ОЗУ, через регистры косвенного адреса R0 и R1 возможен доступ к ВПД емкостью 256 байт.

2.3. Организация ввода/вывода информации

Для связи МК48 с объектом управления, для ввода и вывода информации используются 27 линий. Эти лилии сгруппированы в три порта по восемь линий в каждом и могут быть использованы для ввода или для ввода/вывода через двунаправленные линии. Кроме портов ввода/вывода имеются три линии, сигналы из которых могут изменять ход программы по командам условного перехода: линия ЗПР используется для ввода в МК сигнала запроса прерывания от внешнего источника, линия Т0 используется для ввода тестирующего сигнала от двоичного датчика объекта управлениям, кроме того, под управлением программы (по команде ENT0 CLK) по этой линии может выдаваться сигнал синхронизации, линия Т1 используется для ввода тестирующего сигнала или в качестве входа счетчика событий (по команде STRT CNT).

Порты ввода/вывода Р1 и Р2. Специальная схемотехника портов Р1 и Р2, которая получила название квазидвунаправленной, позволяет выполнять ввод, вывод или ввод/вывод. Каждая линия портов Р1 и Р2 может быть программным путем настроена на ввод, вывод или на работу с двунаправленной линией передачи. Для того чтобы настроить некоторую линию на режим ввода, необходимо в буферный триггер этой линии записать 1. Сигнал СБР автоматически записывает во все линии портов Р1 и Р2 сигнал 1. Квазидвупанравленная структура портов Р1 и Р2 для программиста МК специфична тем, что в процессе ввода информации выполняется операция логического И над вводимыми данными и текущими (последними) выведенными данными. Квазидвунаправленные схемы портов Р1 и Р2 и команды логических операции ANI и ORL предоставляют разработчику эффективное средство маскирования для обработки однобитных входных и выходных переменных.

В системе команд МК есть команды, которые позволяют выполнять запись нулей и единиц в любом разряд или группу разрядов порта, но так как в этих командах маска задается непосредственным операндом, то необходимо знать распределение сбрасываемых и устанавливаемых линий на этапе разработки прикладной программы. В том случае, если маска вычисляется программой и заранее не известна, в ОЗУ необходимо иметь копию состояния порта вывода. Эта копия по командам логических операций объединяется с вычисляемой маской в аккумуляторе и затем загружается в порт. Необходимость этой процедуры вызвана тем, что в МК отсутствует возможность выполнить операцию чтения значений портов Р1 и Р2 для определения прежнего состояния порта вывода. Порт Р2 отличается от порта Р1 тем, что его младшие четыре бита могут быть использованы для расширения МК-системы по вводу/выводу. Через младшую тетраду порта Р2 по специальным командам обращения возможен доступ к че-

тырем внешним четырехбитным портам ввода/вывода P4-P7. Работа этих внешних портов синхронизируется сигналом ПРОГ.

Порт ввода/вывода BUS представляет собой двунаправленный буфер с тремя состояниями и предназначен для побайтного ввода, вывода или ввода/вывода информации. Если порт BUS используется для двунаправленных передач, то обмен информацией через него выполняются по командам MOVX. При выводе байта генерируется стробирующий сигнал ЗП, а выводимый байт фиксируется в буферном регистре. При выводе байта генерируется стробирующий сигнал ЧТ, но вводимый байт в буферном регистре не фиксируется. В отсутствие передач порт BUS по своим выходам находится в высокоимпедансном состоянии. Если порт BUS используется как однонаправленный, то вывод через него выполняется по команде OUTL, а ввод – по команде INS.

Вводимые и выводимые через порт BUS байты можно маскировать с помощью команд ORL и ANL, что позволяет выделять и обрабатывать в байте отдельный бит или группу бит.

В МК-системах простой конфигурации, когда порт BUS не используется в качестве порта-расширителя системы, обмен выполняется по командам INS, OUTL и MOVX. Возможно попеременное использование команд OUTL и MOVX. Однако при этом необходимо помнить, что выводимый по команде OUTL байт фиксируется в буферном регистре парта BUS, а команда MOVX уничтожает содержимое буферного регистра порта BUS. Команда INS не уничтожает содержимое буферного регистра порта. В МК-системах имеющих внешнюю память программ, порт BUS используется для выдачи адреса внешней памяти и для приема команды из внешней памяти программ. Следовательно, в таких системах использование команды OUTL лишено смысла.

2.4. Устройство управления микроконтроллера

Устройство управления МК совместно с логической схемой переходов в каждом цикле команды формирует последовательность сигналов, управляющих функциями всех блоков МК и системой их взаимосвязи. Рассмотрение МК и особенностей реализации тех или иных процедур удобно выполнить путем анализа работы отдельных блоков МК в различных режимах его работы.

Синхронизация МК. Опорную частоту синхронизации определяет или кварцевый резонатор, подключаемый к выводам X1 и X2, или LC-цепь. X1 является входом, а X2 - выходом генератора, способного работать в диапазоне частот от 1 до 6 МГц. На вход X1 может подаваться сигнал от источника внешней синхронизации. Варианты схем синхронизации МК показаны на рис. 6. В состав генератора МК входят два счетчика с модулями пересчета 3 и 5. Первый используется для формирования сигнала системной синхронизации СС (0.5 мкс). Этот же сигнал поступает на счетчик машинных циклов, на выходе которого через каждые пять сигналов синхронизации формируется сигнал САВП (2.5 мкс), идентифицирующий машинный цикл и используемым в расширенных МК-системах для стробирования адреса внешней памяти.

Системный сброс. В обслуживаемых МК-системах для инициализации используется кнопка СБРОС, которая заземляет соответствующий вход МК. В

необслуживаемых МК-системах, к входу СБР подсоединяется конденсатор емкостью 1 мкФ, что обеспечивает подачу сигнала близкого к потенциалу земли, длительностью не менее 50 мкс после того, как напряжение электропитания устанавливается (рис. 7). Сигнал СБР производит следующие действия: сбрасывает счетчик команд и указатель стека, устанавливает порт BUS в высокоимпедансное состояние, а порты P1 и P2 – на режим ввода, выбирает банк регистров 0 и банк памяти 0, запрещает прерывания, останавливает таймер и выдачу синхросигнала на вывод T0, сбрасывает флаг переполнения таймера TF и флаги пользователя F0 и F1.

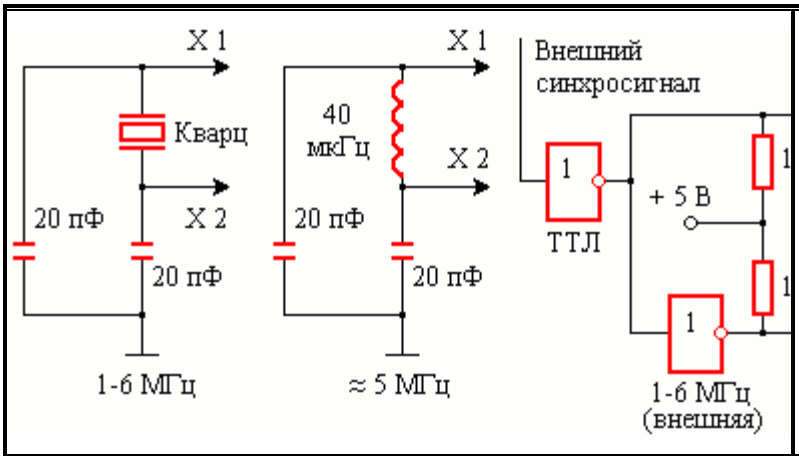

Рис 6. Варианты схем синхронизации МК48

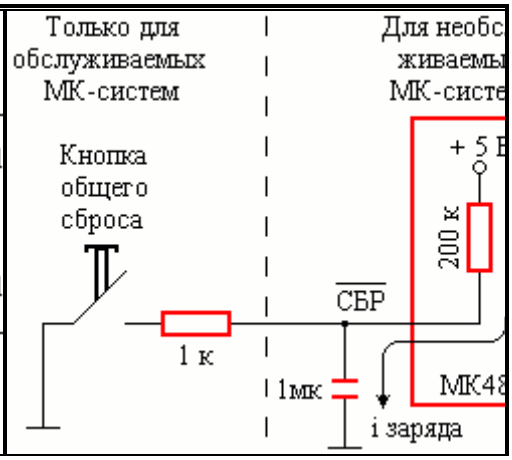

Рис 7. Схема начальной установки МК48

Логика условных переходов. Логическая схема условных переходов МК позволяет программе проверять не только признаки, но и условия, внешние по отношению к МК. По командам условного перехода в случае удовлетворения проверяемого условия в счетчике команд (биты 0-7) из второго байта команды загружается адрес перехода. Логика переходов МК оперирует с набором условий, перечисляемых в таблице 1.

Таблица 1 . Условия переходов по программе

Устройство	Условие перехода инверсное	Условие перехода прямое
Аккумулятор	Не все нули	Все нули
Выбранный бит аккумулятора	-	1
Флаг переноса C	0	1
Флаги пользователя F0 и F1	-	1
Флаг переполнения таймера TF	-	1
Тестовые выходы (T0, T1)	0	1
Вход запроса прерывания ¬ЗПР	0	-

Режим прерывания. Линия запроса прерывания от внешнего источника ЗПР проверяется каждый машинный цикл во время действия сигнала САВП, но

передача управления ячейке 3, где расположена команда JMP, выполняется только по завершению цикла команды. При обработке прерывания, как и при вызове подпрограммы, содержимое счетчика команд и старшей тетрады ССП сохраняется в стеке. К входу ЗПР микроконтроллера через монтажное ИЛИ от схемы с открытым коллектором могут быть подключены несколько источников прерывания. После распознавания прерывания все последующие запросы прерывания игнорируются до тех пор, пока по команде возврат RETR вновь будет разрешена работа логики прерываний. Режим прерываний может быть запрещен или разрешен программам по командам DIS I и EN I. Сигнал ЗПР должен быть снят внешним устройством перед окончанием подпрограммы обслуживания, т.е. до исполнения команды RETR. В том случае, если внешнее устройство не сбрасывает свой флаг запроса прерываний при обращении МК к его буферному регистру, одна из выходных линий МК используется подпрограммой обслуживания прерывания для сброса этого флага во внешнем устройстве. Так как вход ЗПР может быть проверен по команде условного перехода JNI, то при запрещенном режиме вход ЗПР может быть использован в качестве дополнительного тестирующего входа подобноТ0 и Т1.

При необходимости в МК можно создать двухуровневую систему прерываний. Для этого надо разрешить прерывания от таймера, загрузить в него число FFH и перевести в режим подсчета внешних событий, фиксируемых на входе Т1. Переход сигнала на входе Т1 из состояния 1 в состояние 0 приведет к прерыванию по вектору в ячейке 7. В случае одновременного запроса прерываний от внешнего источника и запроса от флага переполнения таймера приоритет остается за источником, воздействующим на вход ЗПР. При входе в подпрограммы обслуживания прерывании старший бит счетчика команд СК11 принудительно устанавливается в нуль. Следовательно, вся процедура обработки прерывания должна быть размещена в банке памяти 0.

Таймер/счетчик. Внутренний 8-битный двоичный суммирующий счетчик может быть использован для формирования временные задержек и для подсчета внешних событий. Содержимое таймера/счетчика (Т/С) можно прочитать (MOV A, T) или изменить (MOV T, A). Две команды STRT T и STRT CNT настраивают и запускают таймер/счетчик в режиме таймера или в режиме счетчика событий соответственно. Остановить работу (но не сбросить содержимое) таймера/счетчика можно или командой STOP TCNT, или сигналом системного сброса СБР. Из максимального состояния FFH таймер/счетчик переходит в начальное состояние 00H. При этом устанавливается в 1 флаг переполнения, который может вызывать прерывание, если оно разрешено командой EN TCNTI. Прерывание от таймера/счетчика может быть запрещено командой DIS TCNTI, но при этом флаг переполнения может быть опрошен по команде условного перехода JTF, как и перевод к подпрограмме обработки прерывания по вектору с адресом 7, сбрасывает флаг переполнения таймера/счетчика.

В режиме таймера на вход таймера/счетчика через делитель частоты на 32 поступают основные синхросигналы машинного цикла САВП (400 КГц), и счетчик увеличивает свое состояние на 1 через каждые 80 мкс (12.5 КГц). Путем программной установки таймера/счетчика в исходное состояние и анализа

флага переполнения могут быть реализованы различные временные задержки, лежащие в диапазоне от 80 мкс до 20.48 мс. Временные задержки, превышающие по длительности 20 мс (256 состояний счетчика), могут быть получены накоплением переполнений в рабочем регистре под управлением программы.

В режиме счетчика событий внутренний счетчик увеличивает свое состояние на 1 каждый раз, когда сигнал на входе T1 переходит из состояния 1 в состояние 0. Минимально возможное время между двумя входными сигналами равно 7.5 мкс (3 машинных цикла при использовании резонатора 6 МГц). Минимальная длительность единичного сигнала на входе T1 составляет 0.5 мкс.

3. Система команд МК48

3.1. Общие сведения о системе команд

Система команд МК48 включает в себя 96 основных команд и ориентирована на реализацию процедур управления. Все команды имеют формат один или два байта (70% команд однобайтные). Время выполнения команд составляет 2.5 мкс или 5.0 мкс (один или дна машинных цикла (МЦ) соответственно) при тактовой частоте 6.0 МГц. Большинство команд выполняется за один машинный цикл. За два машинных цикла выполняются команды с непосредственным операндом, ввода/вывода и передачи управления. Микроконтроллер оперирует с командами четырех типов (рис.8).

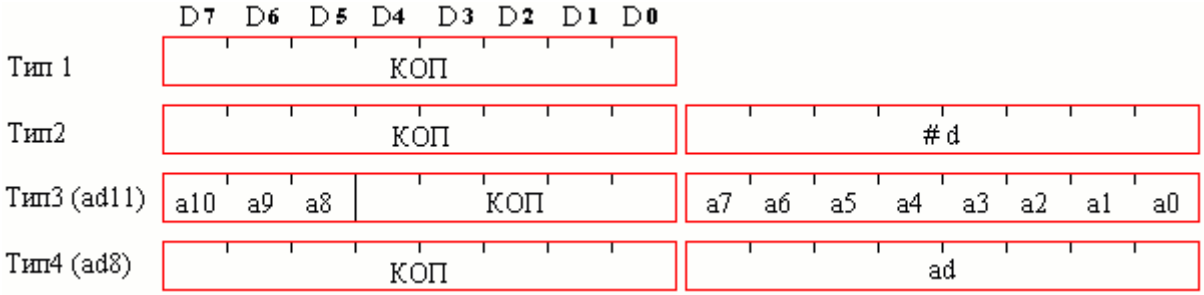

Рис. 8. Типы команд МК48

В МК48 используются четыре способа адресации: прямая, непосредственная, косвенная и неявная.

Все множество команд можно разбить на пять, групп по функциональному признаку: команды пересылки данных, арифметических операции, логических операций, передачи управления и управления режимами работы МК.

К достоинствам системы команд МК48 можно отнести: эффективный ввод/вывод, включая маскирование и возможность управления отдельными битами портов, возможность ветвления по значению отдельных бит, возможность обработки как двоичных, так и десятичных двоично-кодированных чисел.

При выполнении команд могут использоваться значения отдельных флагов, входящих в ССП, и флагов пользователя. Все команды, в результате выполнения которых модифицируются флаги, перечислены в таблице 2.

Ниже приводится краткое описание команд МК48, сгруппированных по функциональному признаку. При описании команд используются мнемокоды языка ассемблера МК48, а операции, выполняемые по командам, описываются на языке микрооператоров с использованием символических имен и сокращений, которые перечисляются в списке сокращений.

Таблица 2. Команды, модифицирующие флаги

Команды	**Флаги**	**Команды**	**Флаги**
ADD,ADDC	C, AC	JTF	TF = 0
CLR C	C = 0	MOV PSW, A	C,AC,F0,BS
CPL C	C	RETR	C,AC,F0,BS
CLR F0	F0 = 0	RLC A	C
CLR F1	F1 = 0	RRC A	C
CPL F0	F0	SEL MB0,SEL MB1	DBF
CPL F1	F1	SEL RB0, SEL RB1	BS
DA A	C, AC		

3.2. Группа команд пересылки данных

Данная группа состоит из 24 команд (табл. 3). Все команды (кроме MOV PSW, A) не оказывают воздействия на флаги. Команды пересылки данных внутри МК выполняются за один машинный цикл, обмен с внешней памятью и портами требует двух машинных циклов.

Структура информационных связей. На рисунке 9 представлен граф возможных пересылок, который иллюстрирует структуру информационных связей МК48. Можно выделить девять типов операндов, между которыми выполняется информационный обмен. Операнды, участвующие в операциях пересылки, различаются по месту расположения и способу адресации. К операндам относятся: аккумулятор, регистры общего назначения, ССП, таймер, порты, непосредственный операнд, внешняя и резидентная память данных и память программ (ПП). Аккумулятор является как бы "почтовым ящиком", через который остальные устройства (операнды) могут обмениваться данными. К памяти программ существует только односторонний доступ для чтения.

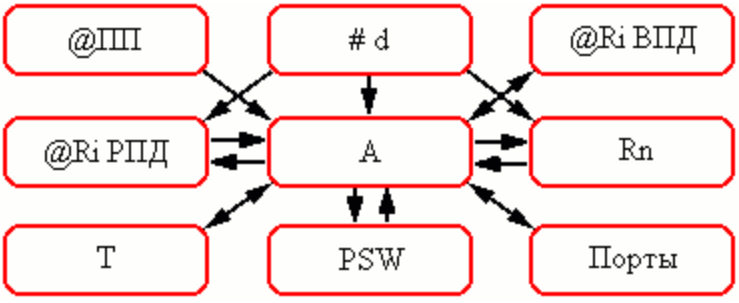

Рис. 9. Граф пересылок данных

Форматы данных. Большинство команд выполняет пересылку 8-битных (1-байтных) операндов. Существуют также несколько команд, оперирующих с 4-битными операндами (тетрадами). Команды пересылки тетрад используются

при обращении к 4-битным портам внешней схемы расширителя ввода/вывода (P4-P7).

Режимы передачи данных. В МК48 возможна передача данных в двух режимах: пересылки (загрузки) и обмена. Пересылка предполагает передачу данных в направлении от источника к приемнику. При этом источник не изменяет своего содержимого. Обмен предполагает одновременную передачу данных в двух направлениях: в результате операции обмена изменяются значения обоих операндов, участвующих в операции.

Таблица 3. Группа команд пересылки данных .
(Т-тип команды, Б-формат в байтах, Ц-число машинных циклов)

Название команды	Мнемокод	КОП	ТБЦ	Операция
Пересылка регистра в аккумулятор	MOV A,Rn	11111rrr	1 1 1	(A) = (Rn)
Пересылка байта из РПД в аккумулятор	MOV A,@Ri	111000i	1 1 1	(A) = ((Ri))
Пересылка непосредственного операнда в аккумулятор	MOV A,#d	00100011	2 2 2	(A) = #d
Пересылка аккумулятора в регистр	MOV Rn,A	10101rrr	1 1 1	(Rn) = (A)
Пересылка непосредственного операнда врегистр	MOV Rn,#d	10111rrr	2 2 2	(Rn) = #d
Пересылка аккумулятора в РПД	MOV @Ri,A	1010000i	1 1 1	((Ri)) = (A)
Пересылка непосредственногооперанда в РПД	MOV @Ri,#d	1011000i	2 2 2	((Ri)) = #d
Пересылка ССП ваккумулятор	MOV A,PSW	11000111	1 1 1	(A) = (PSW)
Пересылка аккумулятора в ССП	MOV PSW,A	11010111	1 1 1	(PSW) = (A)
Пересылка содержимого таймера/счетчика в аккумулятор	MOV A,T	01000010	1 1 1	(A) = (T)
Пересылка аккумулятора в таймер/счетчик	MOV T,A	01100010	1 1 1	(T) = (A)
Пересылка байта из ВПД в аккумулятор	MOVX A,@Ri	1000000i	1 1 2	(A) = ((Ri))
Пересылка аккумулятора в ВПД	MOVX @Ri,A	1001000i	1 1 2	((Ri)) = (A)
Пересылка байта из текущей страницы программной памяти в аккумулятор	MOVP A,@A	10100011	1 1 2	(PC_{0-7}) = (A) (A) = ((PC))
Пересылка байта из третьей страницы программной памяти в аккумулятор	MOVP3 A,@A	11100011	1 1 2	(PC_{0-7}) = (A) (PC_{8-11}) = 0011 (A) = ((PC))
Обмен регистра с аккумулятором	XCH A,Rn	00101rrr	1 1 1	(A)<->(Rn)
Обмен аккумулятора с РПД	XCH A,@Ri	0010000i	1 1 1	(A)<->((Ri))
Обмен младших тетрад аккумулятора и байта РПД	XCHD A,@Ri	0011000i	1 1 1	(A_{0-3})<->$((Ri)_{0-3})$
Пересылка данных из порта Pp (p =1,2) в аккумулятор	IN A,Pp	000010pp	1 1 2	(A) = (Pp)
Стробируемый ввод данных из порта ВЦК	INS A,BUS	00001000	1 1 2	(A) = (BUS)

Пересылка аккумулятора в порт Pp (p=1,2)	**OUTL Pp,A**	**001110pp**	**1 1 2**	**(Pp) = (A)**
Стробируемый вывод данных из аккумулятора в порт BUS	**OUTL BUS, A**	**00000010**	**1 1 2**	**(BUS) = (A)**
Ввод тетрады из порта Pp (p=4-7) схемы расширителя	**MOVD A,Pp**	**000011pp**	**1 1 2**	**(A_{0-3}) = (Pp)**
Вывод тетрады в порт Pp (p = 4 -7) схемы расширителя	**MOVD Pp,A**	**001111pp**	**1 1 2**	**(Pp) = (A_{0-3})**

Способы адресации. Для обращения к данным используются четыре способа адресации:

1)**прямая**, когда адрес операнда содержится в теле самой команды, например:

MOV A, RN ;*(A)<(RN)*

Номер регистра, пересылаемого в аккумулятор, указывается в трех младших битах кода операции (КОП);

2) **непосредственная**, когда сам 8-битный операнд (константа) располагается непосредственно в теле команды (второй байт команды), например:

MOV A, #05 ;*(A)<05*

Содержимое второго байта команды (05) пересылается в аккумулятор;

3) **косвенная**, при которой адрес операнда располагается в регистре (R0 или R1), например:

MOV A, @R0 ;*(A)<((R0))*

Содержимое ячейки РПД по адресу, хранимому в регистре R0, пересылается в аккумулятор;

4) **неявная**, при которой в коде операции содержится неявное (по умолчанию) указание на один из операндов. Чаще всего таким операндом является аккумулятор, как, например, в команде

MOV A, RN ;*(A)<(RN)*

Особенность передач через порты. Поры P1 и P2 представляют собой управляемые буферные регистры. При выдаче информации выводимый байт данных фиксируется в буферном регистре порта, а при вводе он не фиксируется и должен быть прочитан контроллером в течение периода присутствия байта на входах порта. Ориентация МК на применение в устройствах управления объектами, привела к появлению в его системе таких команд, которые позволяют выполнять операции ввода/ вывода информации с использованием маскирования, что предоставляет разработчику удобные средства обмена не только байтами, но отдельными битами и их произвольными комбинациями. Схемотехника портов P1 и P2 такова, что ввод данных в некоторую линию возможен только в том случае, если предварительно в каждый бит порта программой МК была записана 1.

Порт BUS может выполнять все функции, перечисленные для портов P1 и P2, но в отличие от них он не может специфицировать отдельные линии на ввод или вывод. Все восемь линий порта BUS должны одновременно быть либо

входными, либо выходными. По комарам MOVX порт BUS используется в качестве двунаправленного синхронного канала для доступа к ВПД.

3.3. Группа команд арифметических операции

Данная группа состоит из 12 команд (табл. 4) и позволяет выполнять следующие операции над 8-битными целыми двоичными числами без знака: двоичное сложение (ADD), двоичное сложение с учетом переноса (ADDC), десятичная коррекция (DA), инкремент (INC) и декремент (DEC).

При сложении используется неявная адресация источника первого операнда и места назначения результата, в качестве которых выступает аккумулятор. Содержимое аккумулятора А можно сложить с регистром, константой и ячейкой РПД. В результате суммирования возможно появление переноса, который фиксируется в специальном триггере переноса (флаг С). Команда сложения с учетом переноса позволяет выполнять суммирование многобайтных чисел.

При необходимости выполнять двоичное вычитание требуется внести вычитаемое в дополнительный код и произвести сложение с уменьшаемым. Все более сложные операции (умножение, деление) выполняются по подпрограммам.

Таблица 4. Группа команд арифметических операций .
(Т-тип команды, Б-формат в байтах, Ц-число машинных циклов)

Название команды	**Мнемокод**	**КОП**	**Т Б Ц**	**Операция**
Сложение регистра с аккумулятором	ADD A,Rn	01101rrr	1 1 1	(A) = (A)+(Rn)
Сложение байта из РПД с аккумулятором	ADD A,@Ri	0110000 i	1 1 1	(A) = (A)+((Ri))
Сложение константы с аккумулятором	ADD A,#d	0000001 1	2 2 2	(A) = (A)+#d
Сложение регистра с аккумулятором и переносом	ADDC A,Rn	01111rrr	1 1 1	(A) = (A)+(Rn)+(C)
Сложение байта из РПД с аккумулятором и переносом	ADDC A,@ Ri	0111000 i	1 1 1	(A) = (A)+((Ri))+(C)
Сложение константы с аккумулятором и переносом	ADDC A,@d	0001001 1	2 2 2	(A) = (A)+#d+(C)
Десятичная коррекция аккумулятора	DA A	0101011 1	1 1 1	если $((A_{0-3})>9) \vee ((AC)=1)$,то $(A_{0-3}) = (A_{0-3})+6$ затем, если $((A_{4-7})>9) \vee ((C) = 1))$, то $(A_{4-7}) = (A_{4-7})+6$

Инкремент аккумулятора	INC A	00010111	1 1 1	(A) = (A)+l
Инкремент регистра	INC Rn	00011rrr	1 1 1	(Rn) = (Rn)+l
Инкремент байта в РПД	INC @Ri	0001000i	1 1 1	((Ri)) = ((Ri))+l
Декремент аккумулятора	DEC A	00000111	1 1 1	(A) = (A)-l
Декремент регистра	DEC Rn	11001rrr	1 1 1	(Rn) = (Rn)-1

3.4. Группа команд логических операций

Данная группа состоит из 28 команд (табл. 5) и позволяет выполнять следующие операции над байтами: дизъюнкцию, конъюнкцию, исключающее ИЛИ, инверсию, сброс и сдвиг. Две команды (сброс и инверсия) позволяют выполнять операции над битами.

Широко используется неявная адресация аккумулятора в качестве источника операции и места фиксации результата. Вторым операндом в командах может быть регистр, константа или ячейка РПД. Существуют команды (ANL, ORL), оперирующие с портами, что позволяет эффективно управлять значениями отдельных бит при вводе/выводе информации.

Таблица 5. Группа команд логических операций .
(Т-тип команды, Б-формат в байтах, Ц-число машинных циклов)

Название команды	**Мнемокод**	**КОП**	**Т Б Ц**	**Операция**
Логическое И регистра и аккумулятора	ANL A,Rn	01011rrr	1 1 1	(A) = (A)/\(Rn)
Логическое И байта из РПД и аккумулятора	ANL A,@Ri	0101000i	1 1 1	(A) = (A)/\((Ri))
Логическое И константы и аккумулятора	ANL A,#d	01010011	2 2 2	(A) = (A)/\#d
Логическое ИЛИ регистра и аккумулятора	ORL A,Rn	01001rrr	1 1 1	(A) = (A)\/(Rn)
Логическое ИЛИ байта из РПД и аккумулятора	ORL A,@Ri	0100000i	1 1 1	(A) = (A)\/((Ri))
Логическое ИЛИ константы и аккумулятора	ORL A,#d	01000011	2 2 2	(A) = (A)\/#d
Исключающее ИЛИ регистра и аккумулятора	XRL A,Rn	11011rrr	1 1 1	(A) = (A)⊻(Rn)
Исключающее ИЛИ байта из РПД и аккумулятора	XRL A,@Ri	1101100i	1 1 1	(A) = (A)⊻((Ri))
Исключающее ИЛИ константы и аккумулятора	XRL A,#d	11010011	2 2 2	(A) = (A)⊻#d
Сброс аккумулятора	CLR A	00100111	1 1 1	(A) = 0

Инверсия аккумулятора	CPL A	00110111	1 1 1	(A) = (не A)
Обмен тетрад в аккумуляторе	SWAP A	01000111	1 1 1	$(A_{0-3}) <-> (A_{4-7})$
Циклический сдвиг влево аккумулятора	RL A	11100111	1 1 1	$(A_{n+1}) = (An)$, n=0?6 $(A_0) = (A_7)$
Сдвиг влево аккумулятора через перенос	RLC A	11110111	1 1 1	$(An+1) = (An)$, n=0?6 $(A_0) = (C)$; $(C) = (A_7)$
Циклический сдвиг вправо аккумумулятора	RR A	01110111	1 1 1	$(An) = (An+l)$, n=0?6 $(A_7) = (A_0)$
Сдвиг вправо аккумулятора через перенос	RRC A	01100111	1 1 1	$(An) = (An+l)$, n=0?6 $(A_7) = (C)$; (C) =(A_0)
Логическое И константы и порта Pp (p = 1,2)	ANL Pp,#d	100110pp	2 2 2	(Pp) = (Pp)/\#d
Логическое И константы и порта BUS	ANL BUS,#d	10011000	2 2 2	(BUS) = (BUS)/\#d
Логическое И аккумулятора и порта Pp (p=4-7)	ANLD Pp,A	100111pp	1 1 2	$(Pp) = (Pp)/\(A_{0-3})$
Логическое ИЛИ константы и порта BUS	ORL BUS,#d	10001000	2 2 2	(BUS) = (BUS)\/#d
Логическое ИЛИ аккумулятора и порта Pp (p=4-7)	ORLD Pp,A	100011pp	1 1 2	$(Pp) = (Pp)\/(A_{0-3})$
Сброс переноса	CLR C	10010111	1 1 1	(C) = 0
Сброс флага F0	CLR F0	10000101	1 1 1	(F0) = 0
Сброс флага F1	CLR F1	10100101	1 1 1	(F1) = 0
Инверсия переноса	CPL C	10100111	1 1 1	(C) = (неC)
Инверсия флага F0	CPL F0	10010101	1 1 1	(F0) = (не F0)
Инверсия флага F1	CPL F1	10110101	1 1 1	(F1) = (не F1)

3.5. Группа команд передачи управления

Данную группу образуют 19 команд передачи управления, из них две команды безусловного перехода, 14 команд условного перехода, команда вызова подпрограмм и две команды возврата из подпрограмм. В таблице 6 приводится описание команд передачи управления.

Таблица 6. Группа команд передачи управления .
(Т-тип команды, Б-формат в байтах, Ц-число машинных циклов)

Название команды	**Мнемокод**	**КОП**	**Т Б Ц**	**Операция**
Безусповный переход	JMP ad11	$a_{10}a_9a_800100$	3 2 2	(PC_{0-10}) = ad11, (PC_{11}) = DBF
Косвенный переход в текущей странице ПП	JMPP @A	10110011	1 1 2	$(PC_{0-7}) = ((A))$
Декремент регистра и переход,если не нуль	DJNZ Rn,ad	11101rrr	4 2 2	(Rn) = (Rn)-1; если (Rn) не = 0, то (PC_{0-7}) =

				ad, иначе (PC) = (PC)+2
Переход, если перенос	JC ad	11110110	4 2 2	Если (C) = 1, то (PC_{0-7}) = ad, иначе (PC) = (PC) +2
Переход ,если нет переноса	JNC ad	11100110	4 2 2	Если (C)=0, то (PC_{0-7}) = ad, иначе (PC) = (PC) +2
Переход, если аккумулятор содержит нуль	JZ ad	11000110	4 2 2	Если (A)=0, то (PC_{0-7}) = ad, иначе (PC) = (PC) +2
Переход, если аккумулятор содержит не нуль	JNZ ad	10010110	4 2 2	Если (A) не= 0, то(PC_{0-7}) = ad, иначе (PC) = (PC) +2
Переход, если на входе T0 высокий уровень	JT0 ad	00110110	4 2 2	Если T0=1, то (PC_{0-7}) = ad, иначе (PC) = (PC) + 2
Переход, если на входе T0 низкий уровень	JNT0 ad	00100110	4 2 2	Если T0=0, то (PC_{0-7}) = ad, иначе (PC) = (PC) +2
Переход, если на входе T1 высокий уровень	JT1 ad	01010110	4 2 2	Если T1=1, то (PC_{0-7}) = ad, иначе (PC) = (PC) +2
Переход, если на входеT1 низкий уровень	JNT1 ad	01000110	4 2 2	Если T1=0, то (PC_{0-7}) = ad, иначе (PC) = (PC) + 2
Переход, если флаг F0 установлен	JF0 ad	10110110	4 2 2	Если (F0)=1, то (PC_{0-7}) = ad, иначе (PC) = (PC) + 2
Переход, если флаг F1 установлен	JF1 ad	01110110	4 2 2	Если(F1)=1, (PC_{0-7}) = ad, иначе (PC) = (PC) + 2
Переход, если флаг переполнения таймера установлен	JTF ad	00010110	4 2 2	Если TF=1,то TF = 0 , (PC_{0-7}) = ad, иначе (PC) = (PC) + 2
Переход, если на входе ¯ЗПР низкий уровень	JNI ad	10000110	4 2 2	Если ¯ЗПР=0, то (PC_{0-7}) = ad, иначе (PC) = (PC) + 2
Переход, если бит аккумулятора равен единице	JBb ad	bbb10010	4 2 2	Если (Bb) = 1, то (PC_{0-7}) = ad, (b=0 -7) иначе (PC) = (PC) + 2
Вызов подпрограммы	CALL ad11	$a_{10}a_9a_8$10100	3 2 2	((SP)) = (PC),(PSW_{4-7}), (SP) = (SP) + 1, (PC_{11}) = DBF, (PC_{0-10}) = ad11
Возврат из подпрограммы	RET	10000011	1 1 2	(SP) = (SP)-1, (PC) =

				((SP))
Возврат из подпрограммы и восстановление ССП	RETR	10010011	1 1 2	(SP) = (SP) - 1,(PC) = ((SP)), (PSW_{4-7}) = ((SP))

Команды ветвления с прямой адресацией. В большинстве команд прямо указывается адрес перехода. В теле команды при этом содержится 8 (ad) или 11 (ad11) бит адреса перехода. Команда JMP позволяет передать управление в любое место 2048-байтного банка памяти программ (ПП). Номер банка ПП определяется флагом DBF, значение которого копируется в старший бит счетчика команд (PC11) при выполнении команды JMP или CALL. Для перехода из нулевого банка ПП в первый недостаточно только установить флаг DBF, необходимо также выполнить команду перехода JMP, которая изменит значение старшего бита счетчика команд.

Все остальные команды (кроме команд возврата) содержат только восемь младших бит адреса перехода. При этом оказывается возможным осуществить переход только в пределах одной страницы ПП (256 байт).

Если команда короткого перехода расположена на границе двух страниц (т.е. первый байт команды на одной странице, а второй – на следующей), то переход будет выполнен в пределах той страницы, где располагается второй байт команды. Для условного перехода с одной страницы на другую можно воспользоваться тандемом из команды условного перехода и длинного безусловного перехода (JMP).

Переход по косвенному адресу. Команда JMPP осуществляет переход по адресу, содержащемуся в ячейке ПП, на которую указывает содержимое аккумулятора. Таким образом, аккумулятор содержит адрес адреса перехода. Ячейка с адресом перехода должна находиться на той же странице ПП, что и команда перехода JMPP. Команда косвенного перехода обеспечивает простой доступ к таблице, содержащей векторы переходов по программе в зависимости от содержимого аккумулятора, что позволяет легко реализовать механизм множественных ветвлений.

Условные переходы. По командам условных переходов могут проверяться не только внутренние флаги, но и некоторые сигналы на внешних входах МК. Это позволяет эффективно выполнять ветвления в программе без использования процедуры предварительного ввода и последующего сравнения. Все команды условных переходов используют прямую короткую адресацию, что накладывает определенные ограничения на размещение программ в памяти. Анализируемые признаки за исключением C и F0 не фиксируются в специальных триггерах флагов, а представляются мгновенными значениями сигналов в АЛУ или на соответствующих входах МК.

Программные циклы. Для организации цикла удобно использовать команду DJNZ. Счетчик циклов организуется в одном из регистров текущего банка, для чего в этот регистр загружайся число повторений цикла. При выполнении команды DJNZ производится декремент и последующая проверка на нуль содержимого регистра-счетчика циклов. Если его содержимое оказывается не

нулевым, то происходит переход к началу цикла, иначе – выход из цикла. Структура программы при этом будет следующей:

MOV RN, #N; *Инициализация счетчиков цикла*
LOOP; *Тело цикла*
. . . . .
CYCLE: DJNZ RN, LOOP ;*Декремент RN, и переход, если не нуль*

Следует отметить, что команды от метки LOOP до метки CYCLE включительно должны находиться в пределах одной страницы памяти программ.

Работа с подпрограммами. Для вызова подпрограмм существует команда CALL, позволяющая обратиться в любое место текущего банка ПП. При вызове подпрограммы в стеке запоминается адрес возврата и часть ССП. Глубина вложений подпрограмм ограничена емкостью стека (16 байт) и не должна превышать восьми.

Для возврата из подпрограммы необходимо выполнить команду RET, которая восстановит в счетчике команд адрес возврата. Для выхода из подпрограммы обработки прерывания служит команда RETR, которая кроме адреса возврата восстанавливает ССП и разрешает прерывания от данного источника.

При программировании МК необходимо внимательно отслеживать ситуацию вызова подпрограммы, находящуюся в альтернативном (па отношению к текущему) банке ПП. В этом случае перед вызовом подпрограммы необходимо выбрать соответствующий банк памяти, а перед возвратом – восстановить старое значение DBF. Если в подпрограмме нет команды восстановления DBF, то возврат все же будет выполнен правильно (так как в стеке сохранен полный действительный адрес возврата), однако первая же команда длинного перехода передаст управление в альтернативный банк ПП. Если к подпрограмме производятся обращения из разных банков памяти, то оказывается затруднительно восстановить значение DBF перед возвратом. В этом случае можно рекомендовать использовать команду восстановления DBF в основной программе вслед за командой вызова. Пусть, например, требуется вызвать подпрограмму с именем SUBROUT, находящуюся в первом банке ПП, из программы, расположенной в нулевом банке ПП. Корректный вызов подпрограммы реализуется последовательностью команд

```
SEL MB1              ;Установка флага
DBF CALL SUBROUT     ;Вызов подпрограммы (после возврата
                     ; из подпрограммы флаг DBF ;остается равным 1)
SEL MB0              ;Восстановление DBF во избежание
                     ;ошибочного перехода в банк 1
```

3.6. Группа команд управления режимом работы МК

В эту группу входят команды управления таймером/счетчиком, прерываниями и флагами переключения банков регистров и банков ПП. В таблице 7 перечислены команды этой группы.

Таблица 7. Группа команд управления режимами работы МК48.
(Т-тип команды, Б-формат в байтах, Ц-число машинных циклов)

Название команды	Мнемокод	КОП	ТБЦ	ОПЕРАЦИЯ
Запуск таймера	STRT T	01010101	1 1 1	Описание приведено в тексте
Запуск счетчика	STRT CNT	01000101	1 1 1	Описание приведено в тексте
Останов таймера/счетчика	STOP TCNT	01100101	1 1 1	Описание приведено в тексте
Разрешение прерывания от таймера/счетчика	EN TCNTI	00100101	1 1 1	Описание приведено в тексте
Запрещение прерывании от таймера/ счетчика	DIS TCNTI	00110101	1 1 1	Описание приведено в тексте
Разрешение внешнего прерывания	EN I	00000101	1 1 1	Описание приведено в тексте
Запрещение внешнего прерывания	DIS I	00010101	1 1 1	Описание приведено в тексте
Выбор нулевого банка регистров	SEL RB0	11000101	1 1 1	(BS) = 0
Выбор первого банка регистров	SEL RB1	11010101	1 1 1	(BS) = 1
Выбор нулевого банка ПП	SEL MB0	11100101	1 1 1	(DBF) = 0
Выбор первого банка ПП	SEL MB1	11110101	1 1 1	(DBF) = 1
Разрешение выдачи синхросигнала на выход Т0	ENT0 CLC	01110101	1 1 1	Т0-синхросигнал (2МГц)
Холостая команда	NOP	00000000	1 1 1	(PC) = (PC) +1

Операции с таймером. Кроме рассмотренных ранее команд обмена между таймером и аккумулятором (MOV A,T и MOV T,A), по которым содержимое таймера может быть прочитано во время остановки счета и во время счета ("на лету") или изменено (перезагружено), в МК выполняются специальные команды управления режимом работы таймера. Таймер может быть, в зависимости от команды, использован как счетчик тактов или как счетчик событий от внутреннего или внешнего источника сигналов соответственно. Система команд МК располагает средствами разрешения или запрета прерывания от таймера. Специальной командой ENT0 на вывод Т0 разрешается передача импульсов с частотой опорного синхросигнала, поделенной на три. Выдача этого сигнала может быть отключена только сигналом общего сброса. Синхросигнал на выходе Т0 используется для общей синхронизации внешних устройств, согласованных с МК но частоте работы.

Переключение банков регистров и ПП. Переключение банка памяти программ (т.е. изменение старшего бита счетчика команд) происходит в момент выполнения команды длинного перехода или вызова подпрограммы. Наличие команд переключения банков регистров позволяет при вызове подпрограмм и обработке прерываний эффективно использовать второй банк регистров в качестве рабочего, сохраняя параметры вычислительного процесса не в стеке, а в исходном банке регистров. При программировании процедур обработки преры-

ваний можно переключать или не переключать банки регистров. В том случае, когда банки регистров переключаются, возврат к исходному банку регистров будет выполнен автоматически, если подпрограмма обработки прерывания оканчивается командой возврата с восстановлением ССП (RETR).

4. Использование внешней памяти и расширенного ввода/вывода

В тех случаях, когда функционально-логических возможностей однокристального МК оказывается недостаточно, можно относительно простыми средствами расширить МК-систему до следующих размеров: память программ – до 4 Кбайт; память данных – до 320 байт; линии ввода/вывода – практически неограниченно.

Кроме того, путем подключения специализированных БИС, входящих в микропроцессорный комплект К580, в МК-системе могут быть реализованы различные вспомогательные функции: связь с дисплеем и клавиатурой, многоуровневая программируемая система прерываний, сложная система таймирования, связь с телеграфно-телефонными линиями передачи информации и т.д. С использованием средств буферирования и мультиплексирования адресов/данных можно под управлением программы создавать МК-системы любых требуемых состава и назначения.

4.1. МК - системы с внешней памятью программ

Шина BUS по своим свойствам подобна двунаправленной шине данных микропроцессора К580, и все расширения МК выполняются с использованием этой шины. При обращении к резидентной памяти программ МК не генерирует внешних управляющих сигналов (за исключением САВП, который всегда идентифицирует каждый машинный цикл). Начиная с адреса 1024, МК автоматически формирует управляющие сигналы, обеспечивающие выборку команд из внешней памяти. Рисунок 12 иллюстрирует процесс выборки команды из внешней памяти:

- содержимое счетчика команд выводится через порт BUS и младшую тетраду порта P2;
- по срезу сигнала САВП на внешним регистре фиксируется адрес;
- сигналом РВПП разрешается работа внешней памяти;
- по спаду сигнала РВПП шина BUS переходит в режим ввода.

Ячейки памяти с адресами, лежащими за пределами банка памяти 0, могут быть доступны после выполнения команды переключения банков памяти SEL MB1, за которой должна следовать команда перехода (JMP или CALL).
На рисунке 13 показана структура МК-системы с внешней памятью программ.

Три дополнительные БИС памяти емкостью по 1 Кбайту подключаются к шине BUS своими информационными выходами. Младший байт адреса по сигналу САВП фиксируется на внешнем буферном регистре. Старшая тетрада адреса, выводимая через порт P2, не нуждается в буферизации, так как она стабильна на протяжении всего цикла выборки. Два самых старших бита адреса

заводятся на внешний дешифратор (стробируемый сигналом РВПП) для селекции требуемой БИС памяти программ.

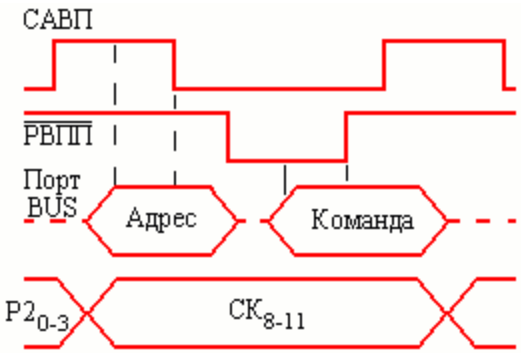

Рис. 12. Временная диаграмма выборки команды из ВПП

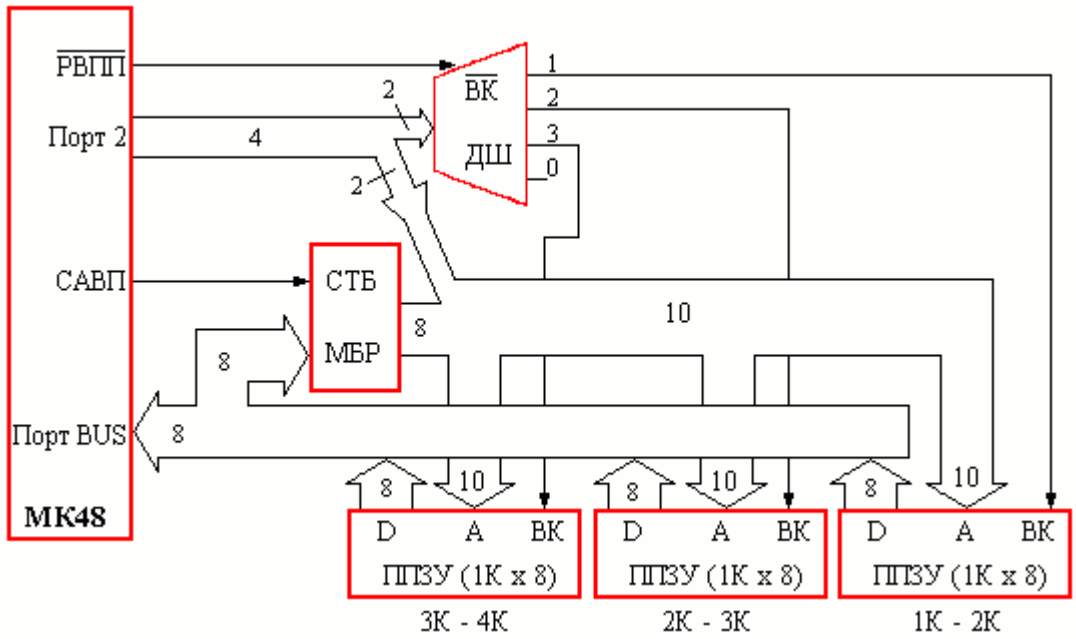

Рис. 13. Структура МК-системы с внешней памятью программ

4.2. МК-система с внешней памятью данных

На рис. 14 показана схема МК-системы, в состав которой включены две дополнительные БИС ОЗУ суммарной емкостью 256 байт.

Внешняя оперативная память доступна МК по командам пересылки MOVX A, @R и MOVX @R,A, которые по косвенному адресу (регистры R0 и R1) выполняют операции передачи байта между ВПД и аккумулятором. Сигналом САВП косвенный адрес, выводимый по шине BUS, фиксируется в многорежимном буферном регистре МБР. Сигналы ЗП и ЧТ определяют режим работы БИС ОЗУ. Так как косвенный адрес имеет формат байта, то схема на рис. 14 обеспечивает адресацию 256 ячеек ОЗУ в дополнение к 64 ячейкам резидентной памяти данных МК48. При необходимости дальнейшего наращивания объема внешнего ОЗУ можно программным способом реализовать механизм "перелистывания" страниц памяти через дополнительные линии порта P2 подобно тому, как это сделано на рис. 13.

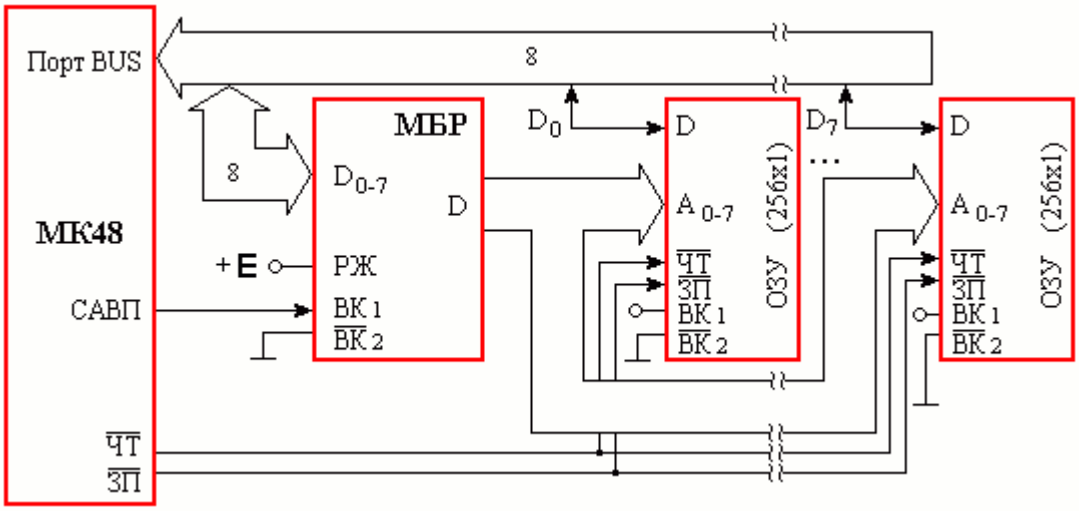

Рис. 14. Структура МК-системы с внешней памятью данных

4.3. МК-система с расширенным вводом/выводом

Для сопряжения МК с объектом, имеющим большое число входов/ выходов, можно расширить резидентную систему ввода/вывода, подключив к МК необходимое количество внешних портов. Такое расширение может быть выполнено двумя способами: с использованием стандартного расширителя ввода/вывода (РВВ) КР580ВР43 или интерфейсных БИС (КР580ВВ55, КР580ВВ51).

Расширитель подключается к МК48 так, как показано на рис. 15 а. Каждый из четырех портов РВВ может использоваться для ввода или вывода информации независимо от остальных, и обеспечивает высокую нагрузочную способность. Для вывода байта данных в порты Р4 и Р5 можно воспользоваться следующей последовательностью команд:

MOVD P4, A *;Вывод A(0...3) в порт 4*
SWAP A *;Обмен тетрад аккумулятора*
MOVD P5, A *;Вывод второй тетрады в порт 5*

На рис. 15 б, в показаны два варианта расширения ввода/вывода с использованием параллельного периферийного адаптера (ППА) КР580ВВ55. В первом варианте (рис. 15 б) порты адаптера адресуются как ячейки ВПД, доступ к которым осуществляется по командам MOVX. Например, для вывода байта в порт В необходимо выполнить две команды:

MOV R1, #1 *;(R1) = Адрес порта B*
MOVX @R1 ,A *;Вывод в порт B из аккумулятора*

Для второго варианта подключения ППА (рис. 15 в), выбор порта осуществляется через установку/сброс двух разрядов порта Р2. Так, для вывода байта данных в порт А можно воспользоваться следующими командами:

ANL P2, #0FCH *;Сброс A0 и A1 адаптера*
OUT BUS, A *;Вывод байта в порт A*

В рассмотренных выше схемах расширения используется только по одной внешней БИС, и поэтому их вход ВК подсоединяется к земле, При необходимости к МК может быть подключено несколько РВВ и ППА.

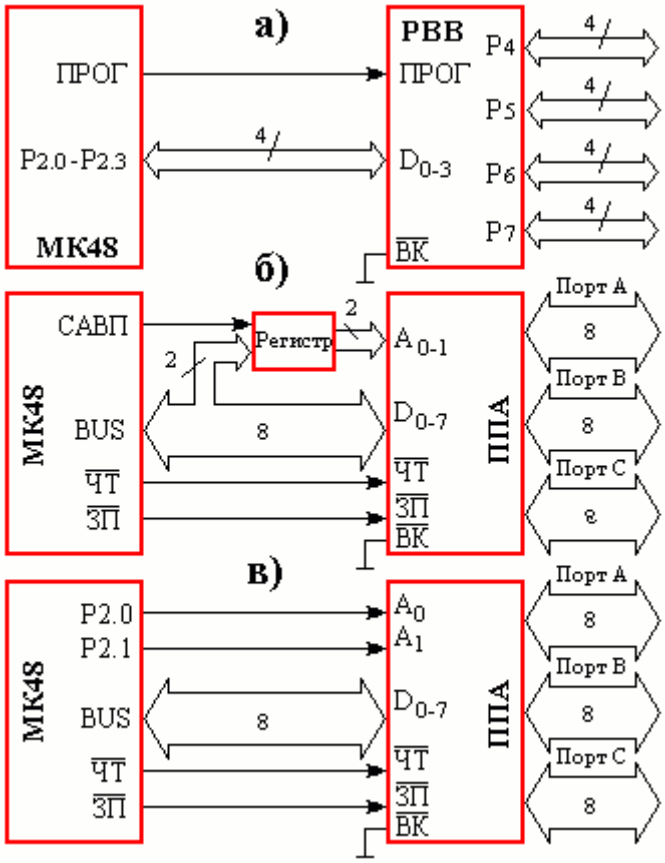

Рис. 15. Варианты расширения ввода/вывода информации в МК-системах

5. Примеры программ обработки данных в МК48

5.1. Примеры использования команд передачи данных

Пример 1. Записать в РПД в ячейки с адресами 41 и 42 число 1C3FH:

```
LOAD:  MOV R0,#41       ; Загрузка в R0 указателя РПД
       MOV @R0,#1CH ; Запись в РПД числа 1CH
       INC R0           ; Продвижение указателя адреса РПД
       MOV  @R0,#3FH; Запись в РПД числа 3FH
```

Пример 2. Переслать текущее содержимое таймера в R5 без потери содержимого аккумулятора:

```
XCHNG: XCH A,R5; Обмен R5 и аккумулятора
       MOV A,T   ; Пересылка содержимого таймера в аккумулятор
       XCH A,R5  ; Обмен R5 и аккумулятора
```

Пример 3. Передать содержимое регистров банка 0 в ВПД, начиная с адреса 50H:

```
       MOV A,#10000B ; Выбор банка регистров 1
       MOV PSW,A
       MOV R0,#50H    ;Определение начального адреса ВПД
       MOV R1,#0      ; Определение начального адреса банка регистров
       MOV R2,#8      ; Счетчик регистров (циклов) <= 8
LOOP:  MOV A,@R1      ; Пересылка байта из регистра в ВПД
       MOVX @R0,A     ; через аккумулятор
```

```
        INC R0          ; Продвижение указателей
        INC R1
        DJNZ R2, LOOP   ; Продолжить, если переданы не все регистры
```

Пример 4. Ввести байт из порта 1 и передать его в порт 2:

```
TRAN:  MOV A, #0FFH    ; Настройка порта 1 на ввод
       OUTL P1, A
       IN A,P1         ; Ввод байта из порта 1
       OUTL P2, A      ; Вывод байта в порта 2
```

Пример 5. Ввести данные из порта P7:

```
INPUT: MOVD A,P7       ; Пересылка четырех битов из порта 7 в
                       ; младшую тетраду аккумулятора
```

Пример 6. Вычислить произведение двух четырех битных чисел, расположенных в младших тетрадах регистров R0 и R1. Для вычисления используется таблица произведений для всех комбинаций сомножителей (всего 256). Произведение двух тетрад имеет формат 1 байта. Таким образом, необходимая таблица произведений занимает одну страницу памяти. Данную страницу удобно разместить на третьей странице РПП:

```
; Вычисление Z=X*Y
; R0=0000.XXXX
; R1=0000.YYYY
; X и Y принимают значения 0 и 1
ORG 0          ; Директива ассемблера, задающий
               ; начальный адрес программы
MOV A,R0       ; Пересылка множимого в аккумулятор
SWAP A         ; Обмен тетрад в аккумуляторе
ORL A,R1       ; Формирование в аккумуляторе адреса
               ; произведения программ
MOVP3 A,@A; Загрузка в аккумулятор произведения
ORG 0300H      ; Директива ассемблера, задающая
               ; начальный адрес таблицы на третьей странице РПП
;Директивы ассемблера, формирующие таблицу произведений
DB 0,0,0,0,0,0,0,0              ; Z=0*Y
DB 0,0,0,0,0,0,0,0
DB 1*0,1*1,1*2,...,1*0FH        ; Z=1*Y
. . . . . .
DB 0FH*0,0FH*1,0FH*2,...,0FH*0FH ; Z=0F*Y
```

Время выполнения программы 12.5 мкс.

5.2. Пример использования команд арифметических операций

Пример 7. Сложить содержимое регистра R7 и ячейки РПД с адресом 60H:

```
MOV R0,#60H    ; Загрузка в R0 адреса РПД
MOV A,R7       ; Загрузка операнда в аккумулятор
ADD A,@R0      ; Сложение
```

Результат операции сложения фиксируется в аккумуляторе, установка флага переноса будет свидетельствовать о переполнении.

Пример 8. Сложить десятичные двоично-кодированные числа (BCD-числа), расположенные в A и R7:

```
ADD A,R7     ; Двоичное сложение
DA A         ; Коррекция результата
```

Пример 9. Инкрементировать содержимое ячеек РПД по адресам 10-18:

```
INCR:  MOV R0,#10          ; Загрузка в R0 начального адреса
       MOV R3,#(18-10+1)   ; Загрузка в R3 числа ячеек
LOOP:  INC @R0             ; Инкремент ячейки РПД
       INC R0              ; Продвижение указателя адреса
       DJNZ R3,LOOP        ; Декремент R3 и повтор, пока R3 не
                           ;равно нулю
```

Пример 10. Сложить много байтные BCD-числа, расположенные в РПД. Регистры R0 и R1 указывают начальные адреса слагаемых. Слагаемые расположены в РПД, начиная с младших байтов. Формат слагаемых одинаков и задается в R2 числом байтов. Результат сложения поместить на место первого слагаемого:

```
; Суммирование Z=W+Y
; (R0) - начальный адрес W
; (R1) - начальный адрес Y
; (R2) - длинна слагаемых W и Y
       CLR C          ; Сброс флага переноса
LOOP:  MOV A,@R0      ; Загрузка текущего байта
       ADDC  A,@R1    ; Сложение
       DA A           ; Коррекция
       MOV @R0, A     ; Размещение текущего байта результата
       INC R0         ; Продвижение указателей байтов
       INC R1         ; слагаемых
       DJNZ R2,LOOP   ; Декремент R2, повтор пока R2 не равно 0
```

Время суммирования составляет (1 + 8N)*2.5 мкс, где N - длина слагаемых в байтах.

Пример 11. Вычитание байтов. Операция вычитания может быть выполнена двумя способами: переводом вычитаемого как отрицательного числа в дополнительный код с последующим сложением; переводом уменьшаемого в обратный код с последующей инверсией суммы. Пусть требуется вычесть из A содержимое регистра R6. Вычитание выполнить в соответствии с выражением

```
(A) <= не ((не (A)) +(R6))
CPL A        ; Инверсия аккумулятора
ADD A,R6     ; Сложение
CPL A        ; Получение разности
```

Установка флага C после выполнения сложения будет свидетельствовать об отрицательном переполнении.

Пример 12. Получить разность 2-байтных числа без знака. Операнды располагаются в РПД. Адрес уменьшаемого храниться в R1, а вычитаемого – в R0. Результат поместить на место уменьшаемого:

```
; Вычислить Z=X-Y
```

```
; X,Y – РПД
; R0 – Адрес Y
; R1 – Адрес X
; Результат на место X
SUBSTR:   MOV A,@R0     ; Загрузка младшего байта Y
          CPL A         ; Получение дополнительного кода Y
          INC A
          ADD A,@R1     ; Вычитание младших байт
          MOV @R0,A     ; Запоминание младшего байта разности
          INC R0        ; Переход к старшим байтам X и Y
          INC R1
          MOV A,@R0     ; Загрузка старшего байта Y
          CPL A         ; Обратный код Y
          ADDC A,@R1    ; Вычитание старших байт
          MOV @R0,A     ; Запоминание результата
```

Пример 13. Умножить однобайтные целые числа без знака. В регистре R1 размещен множитель, в регистре R2 - множимое. Двухбайтный результат умножения будет размещаться в аккумуляторе (старший байт) и в R1 (младший байт) вместо множителя. В регистре R3, выполняющий функции счетчика программных циклов, загружается число 8 (число бит множителя). Умножение выполняется младшими битами вперед со сдвигом вправо частичного произведения. Последовательность действий при этом методе умножения следующая:

1. Содержимое аккумулятора и регистра-расширителя R1 сдвигаются вправо на один бит так, что младший бит множителя, выдвигаемый из регистра R1, помещается в триггер флага C.

2. Если C=1, то множимое добавляется к содержимому аккумулятора, в противном случае никаких операций не производиться.

3. Декрементируется счетчик циклов R3, и если его содержимое не равно нулю , то все действия повторяются.

4. Перед выходом из подпрограммы формируется окончательный результат сдвигом частичного результата на один бит вправо:

```
MPLY1B:   MOV R3,#8; Загрузка счетчика циклов
          CLR A     ; Очистка аккумуляторов
          CLR C     ; Очистка признака переноса
SHIFT:    RRC A     ; Сдвиг аккумулятора вправо
          XCH A,R1  ; Обмен аккумулятора и R1
          RRC A     ; Сдвиг множителя с занесением
                    ; выдвигаемого бита в C
          XCH A,R1  ; Обмен аккумулятора и R1
          JNC RESULT; Если C=1, то суммирование
          ADD A,R2  ; Прибавление множимого
RESULT:   DJNZ R3,SHIFT ; Декремент счетчика и проверка
                    ; окончания операции (R3=0)
          RRC A     ; Сдвиг аккумулятора
          XCH A,R1  ; Обмен
```

```
        RRC A           ; Сдвиг содержимого R1
        XCH A,R1        ; Обмен
```

Максимальное время выполнения программы составляет 200 мкс.

5.3. Примеры использования команд логических операций

Пример 14. Маскирование при вводе. Ввести в регистр R7 информацию из линии 0,1,3,4 и 7 порта 1:

```
IN A,P1             ; Ввод байта из порта 1
ANL A,#10011011B    ; Маскирование
MOV R7,A            ; Передача
```

Пример 15. Ввести в аккумулятор данные из порта 2 и выделить требуемые биты по маске, находящейся в R0:

```
IN A, P2        ; Ввод из порта 2
ANL A, R0       ; Маскирование
```

Пример 16. Выполнить логический сдвиг влево двухбайтного слова, расположенного в (R2) (A):

```
 RLC A          ;Сдвиг младшего байта
 XCH A, R2      ; Обмен аккумулятора и расширителя
 RLC A          ; Сдвиг старшего бита
 XCH A, R2          ; Обмен
```

Пример 17. Выполнить арифметический сдвиг двух байтного слова (R2) (A) вправо:

```
SHIFAR:     CLR C       ; Сброс флага переноса
            CPL C       ; Установка флага переноса
            XCH A, R2   ; Обмен байтами
            JB7 +3      ; Если R2.7 не равно 1, то сброс флага
            CLR C       ; переноса
            RRC A       ; Сдвиг флага переноса в расширитель
            XCH A, R2   ; Обмен
            RRC A       ; Сдвиг младшего байта
```

Пример 18. Умножить аккумулятор на число 2 в степени X, где X - число (не более 8), хранящееся в R2. Умножение на 2 заменяется арифметическим сдвигом влево аккумулятора и расширителя R1:

```
            MOV R1,#0       ; Сброс R1
            CLR C           ; Сброс флага переноса
LOOP:       RLC A           ; Арифметический сдвиг влево
            XCH A,R1        ; объединенного 16-битного результата в
            RLC A           ; регистровой паре (R1) (A)
            XCH A, R1
            DJNZ R2, LOOP   ; Цикл
```

Пример 19. Выдать содержимое аккумулятора в последовательном коде через нулевую линию порта 1, оставляя без изменения остальные биты порта. Передачу вести, начиная с младшего бита:

```
            MOV R1,#8       ; Счетчик бит
LOOP:       JB0 ONE         ; Переход, если бит A.0 равен 1
```

```
        ANL P1,#(NOT 1) ; Сброс P1.0
        JMP NEXT
ONE:    ORL P1,#1       ; Установка P1.0
        JMP NEXT        ; Избыточная команда для выравнивания
                        ;времени передачи 0 и 1
NEXT:   RR A            ; Сдвиг аккумулятора вправо (подготовка
        DJNZ R1, LOOOP ; к передаче очередного бита)
```

Пример 20. Настроить биты 0-3 порта 1 не ввод:

```
        ORL P1, #0FH    ; Установка битов P1.0 ... P1.3
```

Пример 21. Очистить биты 4-7 порта 2:

```
        ANL P2,#0FH     ; Сброс битов P2.4...P2.7
```

Пример 22. Выдать в линию 0 порта 4 значение четвертого бита аккумулятора:

```
        SWAP A          ; Обмен битов 0 и 4 аккумулятора
        ANL A,#1        ; Выделение бита A.0
        ORLD P4,A       ; Установка P4., если A.0=1
        ORL A,#0EH      ; Установка битов 1...3 аккумулятора
        ANLD P4,A       ; Сброс P4.0, если A.0=0
```

Пример 23. Определить четность числа единиц в аккумуляторе:

```
        CLR F0          ; Сброс F0
        MOV R7,#8       ; Число повторов
LOOP:   RRC A           ; Пересылка бита A.0 в перенос
        JNC NEXT        ; Пропустить, если бит равен 0
        CPL F0          ; Подсчет паритета
NEXT:   DJNZ R7,LOOP    ; Повторить 8 раз
```

После выполнения программы аккумулятор сохранит свое значение, флаг F0 будет установлен, если число единиц в аккумуляторе было нечетно. Флаг F0 входит в состав PSW и в данном примере специфицирован пользователем для выполнения функций флага паритета.

5.4. Примеры использования команд передачи управления и команд управления режимом МК48

Пример 24. Передать управление по метке LL, если переключатель банка регистров (бит PSW.4) установлен:

```
JBSET:  MOV A,PSW       ;Передача PSW в аккумулятор
        JB4 LL          ;Переход, если A.4=1
```

Пример 25. Передать управление метке LABEL, если счетчик событий достиг состояния 64:

```
TESTC: MOV A,1          ; Пересылка содержимого счетчика в аккумулятор
        JB6 LABEL       ; Переход пометке, если A.6=1
```

Пример 26. Осуществить переход из нулевого банка ПП к программе с именем ROUT, расположенной в первом банке ПП:

```
        SEL MB1         ; Установка флага DBF
        JMP ROUT        ; Переход к программе ROUNT
```

Пример 27. Множественное ветвление программы.

Допустим, что результатом работы некоторой программы является число X (в пределах от 0 до 15). Необходимо организовать передачу управления 16 различным программам с именами ROUT0 - ROUTF в зависимости от вычисленного значения X:

```
ORG 0          ; Задание начального адреса программы
ANL A,#0FH     ; Сброс старшей тетрады А во избежание ошибки перехода
JMPP @         ; Обращение к таблице векторов переходов
               ;Таблица векторов переходов
DB ROUT0       ; Начальный адрес программы ROUNT0
DB ROUT1       ; Начальный адрес программы ROUNT1
DB ROUT2       ; Начальный адрес программы ROUNT2
          . . .
          . . .
          . . .
DB ROUTF       ; Начальный адрес программы ROUNTF
```

Заметим, что команда JMPP, таблица векторов и программы ROUT0 – ROUTF должны находиться на одной странице ПП.

Пример 28. Организовать ожидание появления нулевого уровня на входе T0:

```
WAIT:  JT1 WAIT   ; Переход на WAIT, если на входе ЗПР ноль
```

Пример 29. Организовать появления единичного уровня на входе ЗПР в предположении, что внешние прерывание запрещены:

```
WAIT:   JT0 WAIT  ; Переход на WAIT, если на входе T0 единица
```

Пример 30. Передача управления одной из восьми программ ROUT0 – ROUT7 при появление нулевого уровня на соответствующем входе порта 1. Наивысшим приоритетом обладает вход P1.0:

```
          ORL P1,#0FFH     ; Настройка порта 1 на ввод
LOOP:     IN A,P1          ; Ввод данных из порта 1
          CPL A            ; Инверсия аккумулятора
          JZ LOOP          ; Ожидание появления первого нуля
          JB0 ROUT0        ; Переход к программе ROUNT0, если P1.0=0
          JB0 ROUT1        ; Переход к программе ROUNT1, если P1.1=0
          JB0 ROUT2        ; Переход к программе ROUNT2, если P1.2=0
          JB0 ROUT3        ; Переход к программе ROUNT3, если P1.3=0
          JB0 ROUT4        ; Переход к программе ROUNT4, если P1.4=0
          JB0 ROUT5        ; Переход к программе ROUNT5, если P1.5=0
          JB0 ROUT6        ; Переход к программе ROUNT6, если P1.6=0
          JB0 ROUT7        ; Переход к программе ROUNT7, если P1.7=0
```

Ветвление осуществляется группой из восьми команд JBb. Приоритеты входов порта 1 определяются очередностью проверки.

Пример 31. При поступление на вход T0 последовательности из восьми нулевых импульсов установить выход P2.7:

```
          MOV R7,#8        ; Загрузка в R7 числа импульсов
ONE:      JT0 ONE          ; Ожидание сигнала 0 на входе T0
```

```
ZERO:  JT0 SKIP          ; Ожидание сигнала 1 на входе T0
       JMP ZERO
SKIP:  DJNZ R7,ONE       ; Повторять, пока не поступит восьмой импульс
       ORL P2,#80H       ; Установка единицы на входе 7 порта 2
```

Длительность нуля и единицы на входе устройства должна быть не менее четырех машинных циклов, т.е. 10 мкс.

Пример 32. Дождаться поступления на вход T0 100 импульсов и перейти по метке PULSE:

```
       MOV A,#156        ; (A)  <== (256 - 100)
       MOV T,A           ; Предустановка счетчика
       STRT CNT          ; Запуск счетчика
WAIT:  JTF PULSE         ; Переход, если прошло 100 импульсов
       JMP WAIT
PULSE: . . .
```

Пример 33. Запретить прерывания от таймера, но разрешить прерывание после восьми сигналов переполнения таймера. При переходе к процедуре обработки прерывания остановить таймер. Сигналы переполнения подсчитывать в регистре 5:

```
STARS: DIS TCNT1         ; Заперт прерываний от таймера
       CLR A             ; Сброс аккумулятора
       MOV T,A           ; Сброс таймера
       MOV R5,A          ; Сброс регистра 5
       STRT T            ; Запуск таймера
M:     JTF COUNT         ; Если TF=1, то переход к COUNT и сброс TF
       JMP M1            ; Цикл
COUNT: INC R5            ; Инкремент регистра 5
       MOV A,R5          ; Пересылка содержимого R5 в аккумулятор
       JB3 INT           ; Переход к подпрограмме обслуживания
                         ; прерывания IN, если бит A.3 равен 1
       JMP M1            ;Переход, если бит A.3 не равен 1
       . . .
       . . .
INT:   STOP TCNT         ; Останов таймера
       JMP 07H           ; Переход к ячейке 7 (вектор прерывания
                         ; от счетчика событий)
```

6. Примеры взаимодействия микроконтроллера с объектом управления

В устройствах и системах логического управления объектами события в объекте управления фиксируются с использованием разнообразных датчиков цифрового и аналогового типов. Наибольшее распространение получили двоичные датчики, например концевые выключатели. Такие датчики соответству-

ют логическому взаимодействию типа да/нет, которое реализуется присутствием логического ноля на линии входа или логической единицы.

6.1 Подсчет числа импульсов между двумя событиями

Часто в управляющих программах возникает необходимость ожидания цепочки событий, представляемой последовательностью импульсных сигналов от датчиков. Предположим, что необходимо подсчитать число деталей, сошедших с конвейера от момента его включения до момента выключения. Факт схода детали с конвейера фиксируется фотоэлементом, на выходе которого формируется импульсный сигнал (рис. 16).

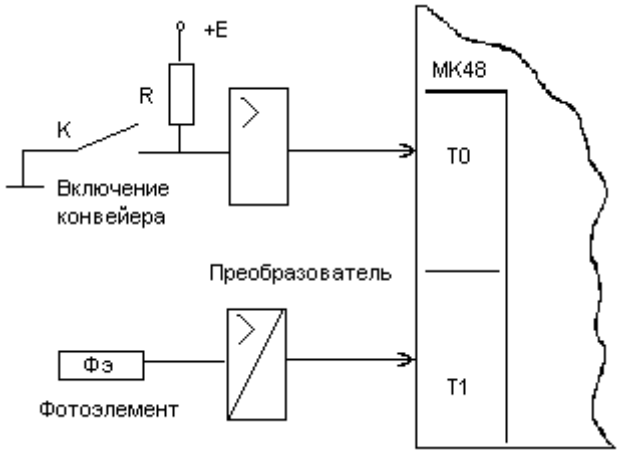

Рис.16 Схема подключения датчиков

Для простоты реализации программы считаем, что общее количество деталей не превышает 255:

```
COUNT:   CLR A        ; сброс счетчика деталей
WAITC1:  JT0 WAITC1   ; ожидание включения конвейера
WAITC2;  JNT1 WAITC2  ; ожидание начала импульса
WAITO2:  JNT1 WAITO2  ; ожидание конца импульса
                INC A          ; инкремент счетчика деталей
                JNT0 WAITC2    ; если конвейер включен, то
                               ; продолжить подсчет деталей
EXIT:
```

По окончании выполнения процедуры в аккумуляторе фиксируется число деталей.

6.2. Опрос группы двоичных датчиков

Микроконтроллеры чаще всего имеют дело не с одним датчиком, как в рассмотренном выше примере, а с группой автономных (логически независимых) или взаимосвязанных (формирующих двоичный код) датчиков (группу взаимосвязанных датчиков называют композицией). При этом МК может вы-

полнять процедуру опроса датчиков и передачи управления отдельным фрагментам прикладной программы в зависимости от принятого кода.

Программную реализацию процедуры ожидания заданного кода (WTCODE) рассмотрим для случая подключения группы из восьми взаимосвязанных статических датчиков к входам порта P1 МК:

```
CODE EQU 10              ; определение эталонного кода
WTCODE: IN A, P1         ; опрос группы датчиков
        XRL A, #CODE     ; сравнение принятого кода
                         ; с заданным кодом CODE
        JNZ WTCODE       ; если коды не совпали, то повторить ввод
EXIT:
```

Сравнение принятого кода с заданным осуществляется операцией "исключающее ИЛИ". В приведенном примере число CODE равно 10.

При опросе композиции двоичных датчиков передачу управления удобно осуществлять по таблице переходов. Ниже приводится текст программы, осуществляющей передачу управления одной из восьми прикладных программ PROG0-PROG7 (которые расположены в пределах одной страницы памяти программ) в зависимости от кодовой комбинации, набранной на переключателях, подключенных к входам P1.0 - P1.2 МК48:

```
GOCODE:  MOV R0, #LOW BASE ; загрузка в R0 начального
                           ; адреса таблицы переходов
         IN A, P1          ; ввод байта
         ANL A, #00000111B ; выделение младших бит
         ADD A, R0         ; формирование адреса строки
                           ; в таблице переходов
         JMPP @A           ; передача управления
BASE:    DB LOW PROG0      ; таблица переходов
         DB LOW PROG1
            . . .
         DB LOW PROG7
```

Программа обеспечивает опрос и выделение сигналов от трех датчиков путем маскирования старших бит аккумулятора. Адрес строки таблицы, в которой хранятся адреса переходов, вычисляется как сумма относительного (внутри текущей страницы РПП) начального адреса таблицы BASE и кода, принятого от датчиков. Команда JMPP @A, таблица BASE и программы PROG0 - PROG7 должны располагаться на одной странице ПП.

Программная реализация цикла ожидания накладывает ограничения на длительность импульса: импульсы длительностью меньше времени выполнения цикла ожидания могут быть "не замечены" МК. Минимально допустимые длительности импульсов для различных способов подключения импульсного датчика к МК приведены в таблице 8.

Таблица 8. Минимально допустимые длительности импульсов для различных способов подключения датчика к МК

Способ подключения датчика к МК48	Минимально допустимая длительность импульса, мкс	
	отрицательного	положительного
P1,P2,BUS/P0	10/2	12,5/2
T0,T1	5/2	5/2
¬ЗПР	10/2	5/2

6.3 . Ввод информации с клавиатуры

Во многих применениях МК работают автономно по заранее заданной программе без вмешательства человека. Наряду с этим существуют интерактивные МК-системы, включающие в контур управления человека-оператора. Простейший пример интерактивной управляющей системы – обслуживаемый МК, требующий ввода оперативной информации и ее отображения. Ввод информации в такой системе может быть организован при помощи клавиатуры.

Для обслуживания клавиатур в МК-системах используются две процедуры: опрос состояния клавиатуры и ввод кода нажатой клавиши. Первая процедура производит однократное обращение к матрице клавиш для определения, нажата ли хотя бы одна из клавиш. Вторая осуществляет циклический опрос клавиатуры до тех пор, пока не будет нажата (а часто и освобождена) клавиша. Будучи встроена в основную программу, вторая процедура блокирует процесс управления объектом на время ожидания нажатия клавиши, а потому обращение к ней осуществляется только при обнаружении нажатой клавиши процедурой опроса состояния клавиатуры.

Процедуру ввода информации с некодирующей матричной клавиатуры удобно рассмотреть на примере клавиатуры 4 * 5, включающей 16 цифровых клавиш (0 - F) и 4 управляющих. Структура матрицы клавиатуры аналогична структуре матрицы двоичных датчиков, способ подключения клавиатуры к МК представлен на рис. 17.

Линии порта 1 используются для сканирования, а линии порта 2 - для опроса матрицы клавиш. Каждая клавиша в такой матрице имеет свой номер, соответствующий ее местоположению. На цифровые клавиши можно нанести обозначения, соответствующие их кодам (от 0 по F) . Коды управляющих клавиш больше числа 15. Диоды обеспечивают защиту от замыкания между собой сканирующих линий в случае одновременного нажатия более чем одной клавиши.

Идентификация нажатой клавиши определяется следующим образом. Каждой клавише клавиатуры должен быть поставлен в соответствие код (ее вес), являющийся функцией номеров линии сканирования (С) и линии возврата (В), на пересечении которых нажата клавиша. Процедура идентификации нажатой клавиши KEYW может быть совмещена с процедурой сканирования. Тогда после выхода из процедуры SCAN в регистре SCANCODE будет размещен код нажатой клавиши.

Для сложных клавиатур SCANCODE не всегда удается совместить с истинным весом клавиши. В этом случае необходима дополнительная перекодировка, которая выполняется табличным способом с использованием SCANCODE в качестве указателя.

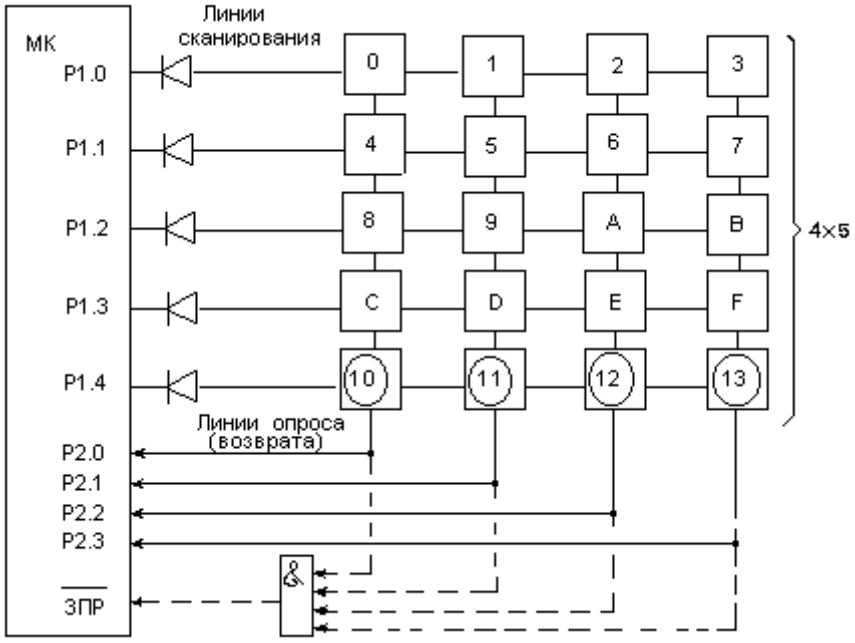

рис. 17 Схема подключения клавиатуры к МК

Процедура KEYW может быть оформлена и как самостоятельная. В этом случае наиболее распространенный способ вычисления веса нажатой клавиши – аналитический в соответствии с выражением W=n*C+B, где n – количество линий возврата. Номера активных линий сканирования и возврата представлены в унитарном коде в виде байта сканирования и байта возврата. Поэтому процедура KEYW реализуется на основе процедур преобразования унитарного кода в двоичный и вычисления W.

Процедура ввода кода клавиши KEYBRD оформляется в виде линейной последовательности рассмотренных выше частных процедур:

```
KEYBRD:
SCAN:                          ; сканирование клавиатуры
            . . .
DBNC:       CALL DALEY         ; задержка для фиксации нажатой клавиши
WAITOP:     . . .              ; ожидание освобождения клавиши
            . . .
DBNC:       CALL DALEY         ; задержка
KEYW:                          ; идентификация нажатой клавиши
            . . .
```

В МК-системах, реализующих непрерывное управление, процедуре KEYBRD должна предшествовать процедура опроса состояния клавиатуры ASK Пример программной реализации процедуры ASK, оформленной в виде подпрограммы, приведен ниже. Выходной параметр передается в основную

программу через флаг переноса C, который устанавливается, если хотя бы одна клавиша нажата:

```
ASK: CLR A      ; сброс аккумулятора
     CLR C      ; сброс флага переноса
     OUTL P1, A      ; вывод байта «все нули» для опроса всех клавиш
     IN A, P2   ; ввод байта возврата
     CPL A      ; инверсия байта возврата
     JZ EXIT    ; выход, если нет нажатой клавиши
     CPL C      ; установка флага переноса
EXIT:     RET        ; возврат из подпрограммы
```

Подпрограмма производит одновременный опрос всех клавиш. В случае, если хотя бы одна клавиша нажата (байт возврата – не все единицы), устанавливается флаг переноса, иначе – сбрасывается.

6.4. Вывод и отображение информации

Многие МК-устройства требуют наличия только простейшей индикации типа да/нет (вкл/выкл). Такая индикация реализуется на основе отдельных светодиодов.

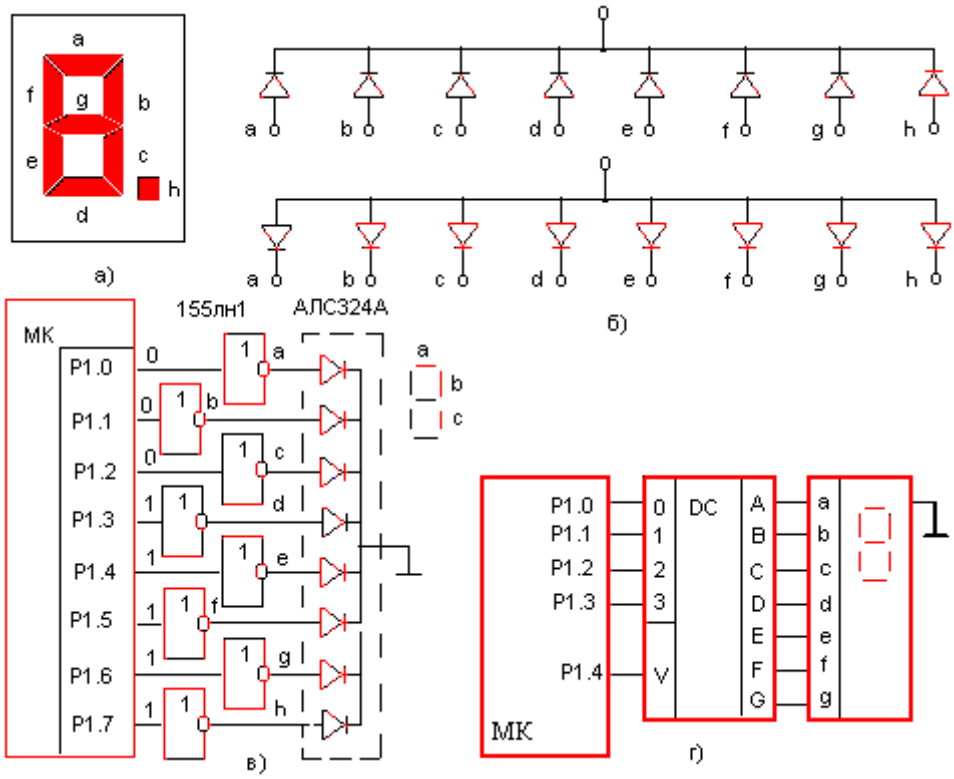

рис. 18 Семисегментный индикатор: а – вид индикатора, б – схемы, в и г – способы подключения к МК

Семисегментные индикаторы (ССИ) широко используются для отображения цифровой и буквенной информации. Семь отображающих элементов позволяют высвечивать десятичные и шестнадцатеричные цифры, некоторые буквы русского и латинского алфавитов, а также некоторые специальные знаки.

Структура ССИ и способы его подключения к МК показаны на рис. 18. Для засветки одного сегмента большинства типов ССИ необходимо обеспечить протекание через сегмент тока 10-15 мА при напряжении 2,0-2,5 В. Низкая нагрузочная способность МК не допускает прямого соединения с ССИ. В качестве промежуточных усилителей тока могут использоваться логические элементы серии К155 или интегральные схемы преобразователей кодов для управления ССИ.

Преобразование двоичных кодов в коды для ССИ может осуществляться либо программно, либо аппаратно с использованием преобразователей К514ИД1, К514ИД2, 133ПП4, 564ИД5.

Матричные светодиодные индикаторы (МСИ) используются для отображения алфавитно-цифровой информации. Каждый из таких МСИ, выполненный в виде интегральной микросхемы, представляет собой матрицу светодиодов размерностью m x n , где n – число колонок, m – число строк матрицы. Наибольшее распространение получили МСИ с размерностью матрицы 7 x 5 и 9 x 7 (рис. 19).

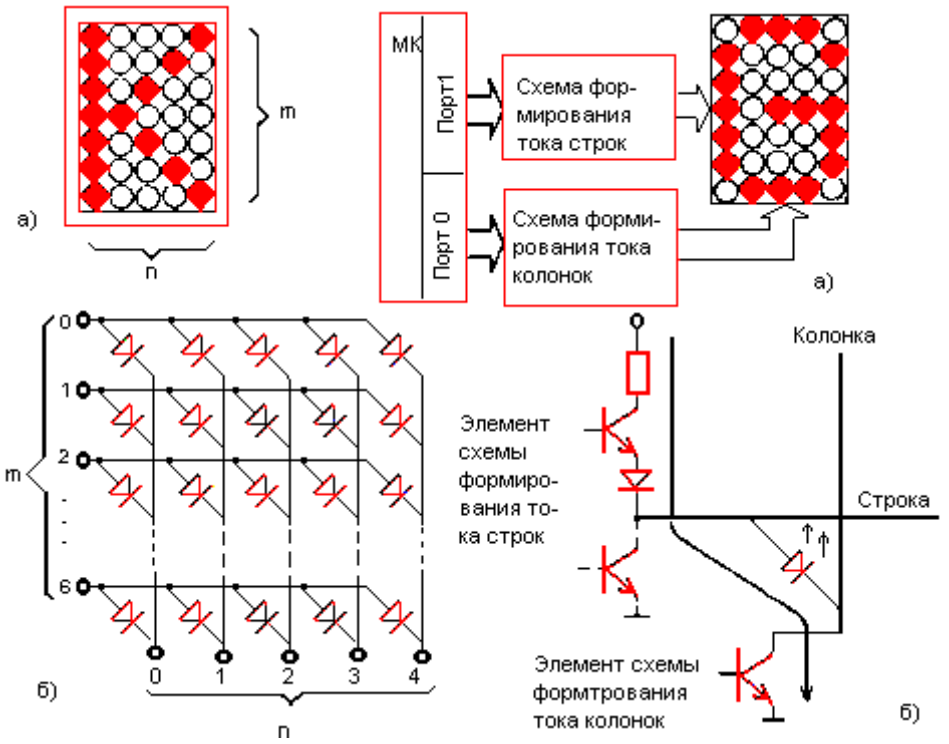

Рис. 19 Матричный индикатор: а – общий вид; б - схема

Рис. 20 Схема подключения матричного индикатора к МК (а) и цепь протекания тока через светодиод (б)

Для включения одного светодиода матрицы необходимо обеспечить протекание через него тока 10-15 мА при напряжении 2,0-2,5 В. Подключение матричного индикатора к МК осуществляется через управляемые схемы формирования тока колонок и строк (рис. 20). Для отображения многосимвольной информации используются линейные (однострочные) дисплеи**.** Такие дисплеи представляют собой "линейку", смонтированную из отдельных ССИ или МСИ.

Число знакомест дисплея определяется в соответствии с требованиями к МК-системе.

Существует два способа организации интерфейса МК с линейным дисплеем: статический и динамический.

Первый требует наличия на входах каждого индикатора специальных буферных регистров для хранения кодов выводимых символов. Естественно, что с увеличением разрядности дисплея возрастает число дополнительных микросхем, а следовательно, и стоимость МК-системы.

Второй способ (динамический) основан на том, что любой световой индикатор является инерционным прибором, а человеческому глазу отображаемая на дисплее информация, если ее обновлять с частотой примерно 20 раз в секунду, представляется неизменяемой. Динамический способ вывода информации на дисплей требует значительно меньших аппаратурных затрат, но более сложного программного обеспечения. Именно этот способ организации ввода информации получил преимущественное распространение в МК-системах.

При использовании внешних (по отношению к МК) схем преобразователей кодов процедура индикации одного символа сводится к выдаче двоичного кода символа в соответствующий порт вывода МК.

При программной перекодировке МК должен поставить в соответствие двоичному коду индицируемого символа определенный байт индикации (БИ), который и выдается в выходной порт. Перекодировку удобнее всего проводить табличным способом. Для этого байты индикации размещаются в смежных ячейках РПП в порядке возрастания исходных двоичных кодов символов. Такое расположение байтов индикации упрощает процесс перекодировки, так как в этом случае требуемый байт находится в строке таблицы с номером, равным двоичному коду индицируемого символа:

```
SYMBOL:  MOV A, @R0          ; загрузка кода символа
         ADD A, #CODTBL ; формирование адреса байта индикации
         MOVP3 A, @A         ; считывание байта индикации из таблицы
         OUTL P1, A          ; выдача байта индикации в порт P1
```

Приведенный фрагмент программы рассчитан на то, что гашение ССИ осуществляется при инициализации системы. Одновременно с этим в регистре R0 формируется адрес ячейки CODE, в которой хранится двоичный код символа:

```
INIT:    . . .
         ORL P2, #0FFH     ; гашение индикатора
         MOV R0, #CODE  ; загрузка в R0 адреса CODE
         . . .
```

Лабораторная работа 1.
Знакомство с отладчиком программ для МК48.

Программа SCM (Single-Chip Machine) выполнена в виде независимого запускаемого модуля, работоспособного под управлением операционной системы MS Window 95/98/2000/NT/XP. SCM включает средства отладки и редактирования программ на ассемблере со встроенным интерпретатором.

Система моделирования SCM предназначена для следующего:

- моделирования работы микро-ЭВМ КМ1816ВЕ48 в совокупности с микросхемой-расширителем портов ввода вывода КР580ВР43 и блоком внешней памяти данных объёмом 256 байт;
- разработки и отладки программ для микроконтроллеров серии МК48;
- исследования поведения внутренних и внешних сигналов указанных микросхем.

Выполнение программы пользователя осуществляется с максимальным приближением к действительности с помощью имитационной модели, уровень детализации которой равен одному такту (1т=0.5 мкс). Доступны следующие режимы моделирования:

- на один такт вперед;
- на один машинный цикл вперед;
- на один шаг вперед;
- выполнение шага до изменения регистра адреса микроконтроллера;
- выполнение до ближайшей точки останова;
- выполнение до конца программы;
- выполнение до первой пустой ячейки памяти;
- на один машинный цикл назад;
- на один такт назад.

Кроме того, пользователю предоставляется такие средства, как:

- временные диаграммы внутренних и внешних сигналов;
- имитация внешних сигналов с отображением изменений на условно-графическом отображении микросхем;
- возможность изменения значений узлов микро-ЭВМ в процессе работы модели и др.

Встроенный редактор-компилятор позволяет набирать программы на ассемблере МК48, а затем с помощью кнопки “компиляция” перевести текст программ в машинные коды и записать его, как в файл ПЗУ с расширением “.MPM”, так и в ПЗУ микроконтроллера для отображения в отладчике.

Кроме того поддерживаются следующие функции распределенного моделирования (на нескольких компьютерах):

- приём библиотек
- загрузка системы команд

- сопряжения с другими программами-эмуляторами;
- обмена сообщений между пользователями, подключенными к одному серверу.

Определим основные рабочие зоны программы отладчика (рис.16):

- главное меню программы;
- кнопочная панель инструментов;
- память программ;
- оперативная память;
- порты микроконтроллера;
- АЛУ;
- слово состояния микроконтроллера (PSW).

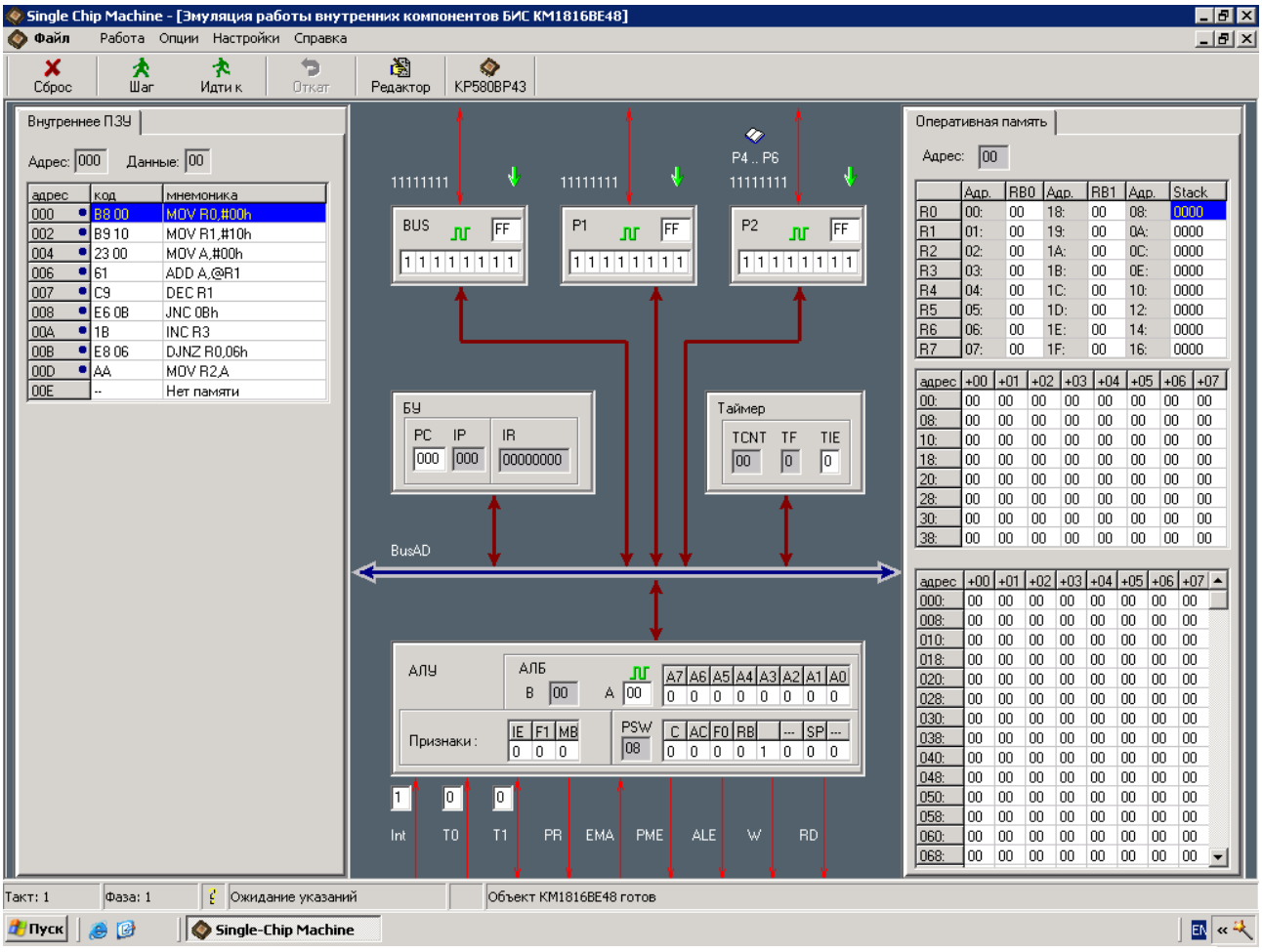

Рис.16 Основные зоны отладчика.

Для набора программы воспользуйтесь кнопкой в панели инструментов «редактор». После ввода программы не забудьте сохранить вашу программу. По умолчанию, программа сохраняется в преопределенную папку ./SCM/Source/noname.asm. Появление ошибок при наборе программы сопровождается сообщением об ошибке и красным квадратиком в строке состояния окна редактора.

Цель лабораторной работы заключается в ознакомлении с основными компонентами отладчика SCM. Используя примеры программ, рассмотренные

в курсе лекционного материала, наберите и проверьте правильность их работы. А так же, при отладке этих программ, рассмотрите работу каждого элемента микроконтроллера в отладчике. В данной работе не только важно вспомнить особенности микроконтроллера МК48, но познакомиться с возможностями программы SCM.

Лабораторная работа 2.
Изучение приемов сложения чисел.

В разделе «Система команд МК48» представлены команды сложения типа ADD и ADDC. Все команды имеют в качестве первого операнда аккумулятор (A), вторым операндом является либо регистр (R0...R7), либо константа #D, либо любая ячейка резидентной памяти данных (РПД). Результат выполнения команды сложения всегда остается в аккумуляторе. Примеры сложения однобайтных чисел могут быть следующими:

- ADD A, R5 ; Сложение содержимого A и R5
- ADD A, #203 ; Сложение A с константой
- MOV R0, #36 ; Пересылка числа в регистр R0
  ADD A, @R0 ; Сложение A с содержимым ячейки РПД

Сложение чисел, размер которых более одного байта, производится побайтно, начиная с младшего байта. При сложении второго и всех последующих байтов необходимо учитывать наличие или отсутствие переноса, который фиксируется во флаге C. Поэтому для сложения второго и последующего байтов необходимо использовать команду ADDC. Приведем числовой пример сложения двух двухбайтных чисел. Пусть первое число A равно 695. Второе число B равно 344. Отсюда старший байт первого числа равен 2, младший байт первого числа равен 183, то есть 695 = 256*2 + 183. Старший байт второго числа равен 1, младший байт второго числа равен 88, то есть 344 = 256 * 1 + 88. Сложение младших байтов чисел 183 + 88 = 271 дает результат, превышающий размер байта, поэтому результатом операции будет число 271 - 256 = 15, а флаг C установится в единичное состояние (C = 1). Старшие байты складываются с учетом флага переноса и дают результат 2 + 1 + 1 = 4. В результате получаем двухбайтное число 4*256 + 15 = 1039. Проверка 695 + 344 = 1039.

Пример программы сложения двух двухбайтных чисел, которые расположены в РПД: первое – по адресу 20H (мл. байт), 21H (ст. байт);
второе – по адресу 30H (мл. байт), 31H (ст. байт). Результат сохраняется на месте второго слагаемого. Ниже представлен пример программы:

```
MOV R0, #20H    ; загрузка младшего
MOV A, @R0      ; байта первого числа в A.
MOV R1, #30H    ; сложение A с младшим
ADD A, @R1      ; байтом второго числа
MOV @R1, A      ; сохранение младшего байта результата
INC R0          ; адресация
INC R1          ; старших байтов
```

```
MOV A, @R0      ; загрузка ст. байта первого числа в А
ADDC A, @R1     ; сложение А со ст. байтом второго числа
MOV @R1, A      ; сохранение ст. байта результата
.END            ; директива конца программы
```

Задача 2.1. Исследовать программу в среде отладчика и заполнить следующую таблицу

Таблица 2.1. Исходные данные для задачи 2.1.

Первое число			Второе число			Результат			
Младший байт	Старший байт	Десятичное представление	Младший байт	Старший байт	Десятичное представление	Младший байт	Старший байт	Состояние флага С	Десятичное представление
		11559			3150				
		15575			3888				
		24336			51590				
		2D27H			0C4EH				
		3D08H			0F30H				
		5611H			CA86H				
117		39245		39	15639				
	56H	7813H	13H	ABH					
73		23ACH		15	2B3DH				
15H	29		17		52639				

Разберем программу суммирования содержимого всех ячеек внешней памяти данных (ВПД). Так как число всех ячеек ВПД равно 256, максимальное значение суммы может быть равно 256 * 255 = 65280 (> 256). Тогда под результат нужно зарезервировать 2 байта в памяти. В нашем примере результат помещается в регистры R2 – ст. байт, R1 – мл. байт. Алгоритм для данной задачи представлен на рис.17, а программа представлена ниже:

```
; R0 – адрес текущей ячейки ВПД
; R1 – мл. байт суммы
; R2 – ст. байт суммы
        MOV R0,#0
        MOV R1,#0
        MOV R2,#0
M1:     MOVX A, @R0
        ADD A, Rl
        MOV R1, A
        JNC R1
        INC R2
M2:     DJNZ R0, M1
        .END
```

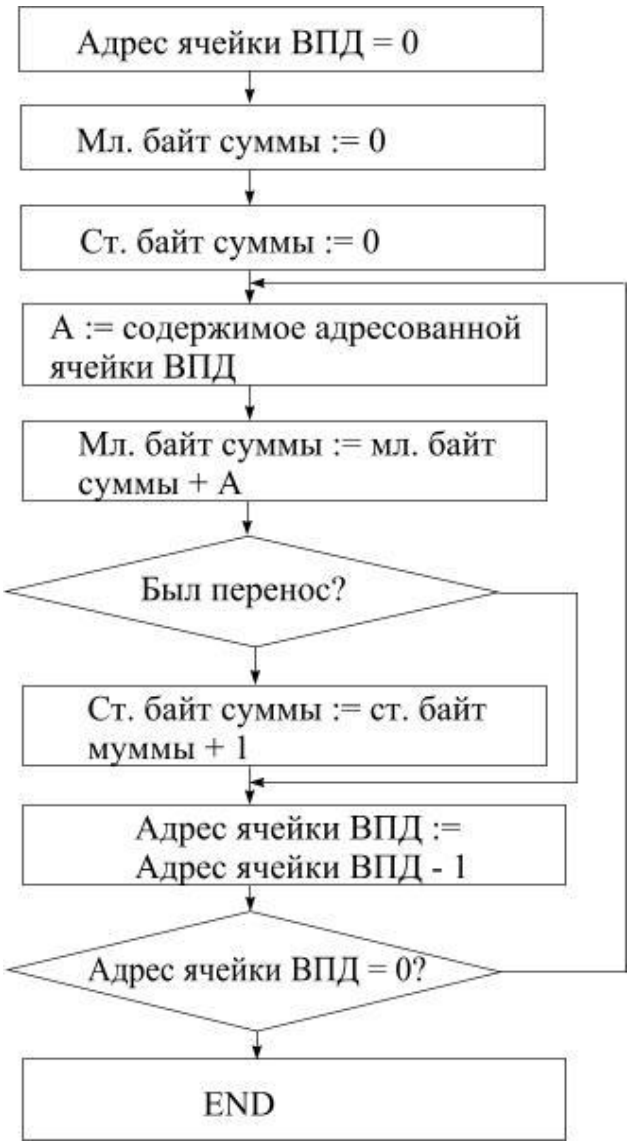

Рис.17 Алгоритм сложения всех ячеек РПД

Задача 2.2. Исследовать программу в среде отладчика и заполнить таблицу 2.2

Таблица 2.2. Исходные данные для задачи 2.2.

Ячейка ВПД, которые нужно заполнить, остальные ячейки равны 0										Сумма
00H	15H	29H	4AH	93H	A2H	B1H	C9H	F3H	FFH	
36H	A1H	12H	9CH	B4H	3DH	E5H	C2H	1BH	73H	
16H	6AH	83H	B9H	13H	29H	99H	A1H	0EH	17H	

Лабораторная работа 3.
Изучение приемов вычитания чисел.

Для вычитания одного числа из другого в системе команд МК48 нет отдельной команды, но это не значит, что вычитание нельзя реализовать. Рассмотрим вычитание двух однобайтных чисел. Если речь идет о вычитании, то

необходимо учитывать знак числа, то есть число должно быть представлено в формате со знаком. Как известно, положительные числа представляются своим прямым кодом, а отрицательные – дополнительным кодом. Признаком того, что число является отрицательным является наличие единицы в старшем (знаковом) разряде числа. Например, число 255 = FFH =11111111B – число отрицательное, значит оно представлено в дополнительном коде. Переведем число из дополнительного кода в прямой, чтобы определить абсолютную величину числа.

$X_{обр.} = X_{доп} – 1$

$X_{обр} = 11111111 – 1 = 11111110$

$X_{пр}$ = инверсия от $X_{обр}$

$X_{пр} = 00000001$

Отсюда запись FFH означает отрицательно число –1.

Исходя из вышеизложенного, можно сделать вывод: чтобы вычесть число B из числа A можно к A прибавить дополнительный код числа B. Пусть число расположено в регистре R2, а число A в регистре R3. Программа вычитания выглядит следующим образом:

```
MOV A, R3   ; поместить вычитаемое в аккумулятор
CPL A       ; инвертировать вычитаемое
INC A       ; получение дополнительного кода вычитаемого
ADD A, R2   ; суммирование уменьшаемого с дополнительным
.END        ; кодом вычитаемого.
```

Задача 3.1. Исследовать программу в среде отладчика и заполнить таблицу 3.1

Таблица 3.1. Исходные данные для задачи 3.1.

Уменьшаемое (R2)	Вычитаемое (R3)	Состояние флага C	Результат (A) в шестнадцатеричном виде
3	2		
2	3		
6	6		
5	0		
0	5		
29	-12		
29	-35		
-36	79		
-75	24		
-7	-7		
-13	-5		
-15	-56		

Задача 3.2. Ответить на вопрос: какой вывод можно сделать об относительной величине чисел R2 и R3 по значению флага C в результате выполнения программы?

Рассмотрим представление двухбайтного числа со знаком. Положительное двухбайтное число представляется своим прямым кодом, знаковый разряд 15 число равен 0. Отрицательное число представляется дополнительным кодом, знаковый разряд 15 числа равен 1.

Задача 3.3. Написать программу вычитания двух двухбайтных чисел, представленных в формате со знаком.

Лабораторная работа 4.
Изучение приемов сравнения чисел.

При сравнении двух числе могут быть поставлены два вопроса:

- равны ли числа друг другу?
- какое из двух чисел больше?

Для ответа на первый вопрос удобно использовать логическую команду исключающее ИЛИ (сравнение по модулю 2). Команда имеет мнемонику XRL и существует в 3-х модификациях.

- XRL A, Rn
- XRL A, #D
- XRL A, @Ri

Из таблицы истинности для данной функции следует, что если операнды, участвующие в операции равны друг другу, то результат операции равен 0.

Следующая программа сравнивает два числа в регистрах R0 и R1.

```
MOV A, R0
XRL A, R1
.END
```

Если в аккумуляторе останется 0, то числа R0 и R1 равны друг другу.

Задача 4.1. Написать программу, которая производит поиск числа 56 в регистрах R2, R3, R4, R5. В регистре R1 остается 0, если искомое число не найдено или номер регистра, в котором обнаружено искомое число.

Если требуется ответить на вопрос какое из двух чисел, например в регистре R2 и R3 больше другого, то необходимо использовать вычитание чисел.

Задача 4.2 Используя выводы задачи 3.1 написать программу сравнения 2-х однобайтных чисел, представленных в форме со знаком, находящихся в регистрах R2 и R3. Результат оставить в регистре R0 так:

R1 = 0, когда R2 = R3
R1 = 1, когда R2>R3
R1=2, когда R2<R3.

Лабораторная работа 5.
Изучение приемов работы с памятью данных.

В качестве памяти данных будем рассматривать внешнюю память данных (ВПД). Выделим для рассмотрения 3 задачи:

- заполнение области ВПД константой;
- поиск чисел в ВПД;
- упорядочение данных в ВПД.

При работе с памятью выделяют три типа адресации:

- непосредственная адресация – операнд записывается в теле самой команды, например: MOV A, #55H [(A) <- 55H];
- прямая адресация – адрес команды записывается в теле самой команды, например: MOV A, R4 [(A) <- (R4)];
- косвенная адресация – в теле команды находится адрес ячейки памяти, содержащей операнд, например: MOV A, @R0 [(A) <- @(R0)].

Следующая программа заполняет константой DAH всю область ВПД

```
        MOV R0, #FFH
        MOV A, #DAH
M0:     MOVX @R0, A
        DJNZ R0, M0
```

Задача 5.1 Написать программу заполнения константой 59H области ВПД с начальным адресом 0H и конечным 35H.

Следующая программа производит поиск числа 59H во всей ВПД. Результат (количество найденных чисел) сохраняется в регистре R0.

```
        MOV R0, #0
        MOV R1, #0
M1:     MOVX A, @R1
        XRL A, #59H
        JNZ M0
        JNZ M0
        INC R0
M0:     DJNZ R1, M1
        .END
```

Задача 5.2 Используя программу задачи 4.2, написать программу поиска максимального из всех чисел ВПД, представленных в формате со знаком. Программа задачи 4.2 должна входить в данную программу в виде подпрограммы.

Задача 5.3 Используя программу задачи 4.2, написать программу упорядочения данных в ВПД в порядке возрастания. Программа должна переставить числа в ВПД чтобы для любой пары чисел в ВПД соблюдалось условие: число, имеющее больший адрес – больше.

Лабораторная работа 6.
Изучение приемов умножения чисел.

Рассмотрим умножение 2-х однобайтных чисел, представленных без знака. Умножить число A на число B – это равносильно тому, что число A просуммировать B раз или число B просуммировать A раз. Очевидно, что результат умножения двух однобайтных чисел будет иметь размер не более двух байтов. Доказать самостоятельно. Пусть первое число хранится в регистре R0, второе – в регистре R1. Результат сохраняется: в регистре R3 – старший байт, в регистре R2 – младший байт. Умножение осуществляется суммированием содержимого регистра R1 столько раз, сколько содержится в регистре R0. Блок-схема алгоритма представлена ниже.

Задача 6.1. Написать программу, реализующую данный алгоритм. Ответить на вопрос: зависит ли время выполнения программы от входных параметров?

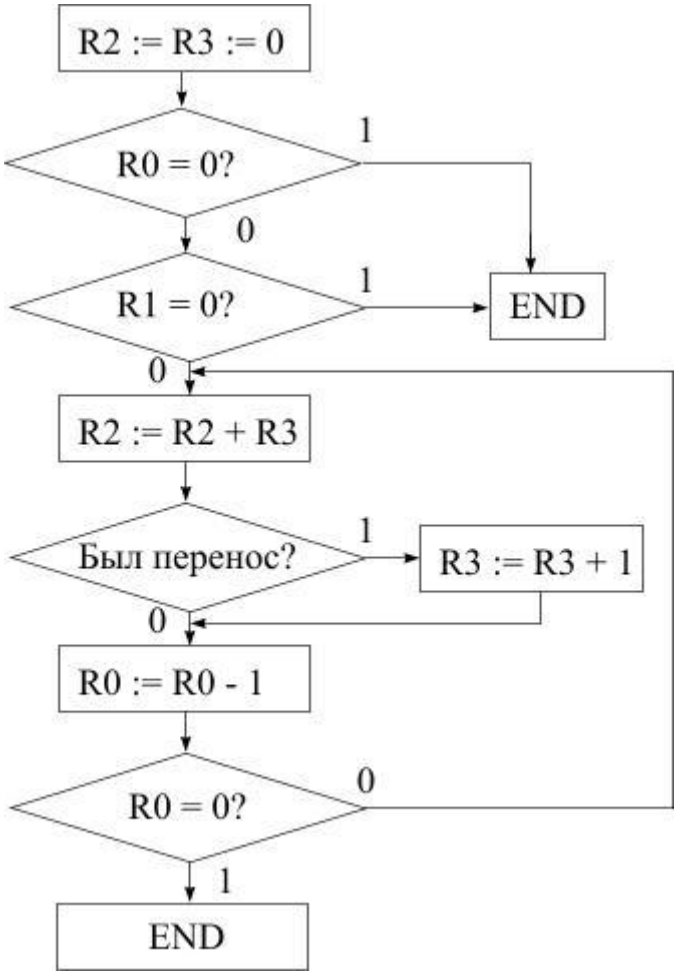

Рис. 18 Алгоритм умножения чисел R0 и R1

Второй вариант умножения – это использование команд суммирования и сдвига. Рассмотрим конкретный пример умножения двух чисел A и B.

A = 10011110 и B = 01010110. Умножим столбиком число A на число B.

Номер						A	=	1	0	0	1	1	1	1	0
Строки						B	=	0	1	0	1	0	1	1	0
0								0	0	0	0	0	0	0	0
1							1	0	0	1	1	1	1	0	
2						1	0	0	1	1	1	1	0		
3					0	0	0	0	0	0	0	0			
4				1	0	0	1	1	1	1	0				
5			0	0	0	0	0	0	0	0					
6		1	0	0	1	1	1	1	0						
7	0	0	0	0	0	0	0	0							
Результат	0	1	1	0	1	0	1	0	0	0	1	0	1	0	0

Из примера видно, что результат получается суммированием 8 строк, сдвинутых на одну относительно другой влево. Содержимое каждой строки равно либо 0 либо равно числу A. Заметим, что содержимое строки равно 0, если соответствующий разряд числа B равен нулю. В нашем случае – это разряды 0, 3, 5, 7. Вариант программы, реализующий умножение методом сложения и сдвига представлен ниже. Здесь R2, R3 – регистры результата, R7 – счетчик сдвигов, R4 – промежуточный буфер для хранения значения текущего старшего байта результата, R0 и R1 –исходные числа.

```
        MOV R2, #0
        MOV R3, #0
        MOV R4, #0
        MOV R7, #8
M3:     MOV A, R1
        RRC A
        MOV R1, A
        JNC M0
        MOV A, R2
        ADD A, R0
        MOV R2, A
        JNC M1
        INC R3
M1:     MOV A, R3
        ADD A, R4
        MOV R3, A
M0:     CLR C
        MOV A, R0
        RLC A
        MOV R0, A
        MOV A, R4
        RLC A
        MOV R4, A
        .END
```

Задача 6.2 По представленному тексту программы восстановить блок-схему алгоритма.

Лабораторная работа 7.
Изучение приемов деления чисел.

Рассмотрим деление чисел, представленных без знака. Результат арифметического деления в общем случае – число дробное. Для простоты рассмотрим целочисленное деление. При целочисленном делении результат представляется в виде двух чисел: целой части числа и остатка от деления. Операцию нахождения целой части числа называют дивергенцией и обозначают DIV. Например A DIV B – целая часть от деления числа A на число B (27 DIV 4 = 6).

Остаток от деления одного числа на другое называется модулем, обозначается MOD. Например A MOD B – остаток от деления числа A на число B (27 MOD 4 = 3).

Нижеприведенный алгоритм реализует нахождение модуля и целой части от деления числа в регистре R1 на число в регистре R2. Результат сохраняется следующим образом:

R0 := R1 DIV R2
R1 := R1 MOD R2

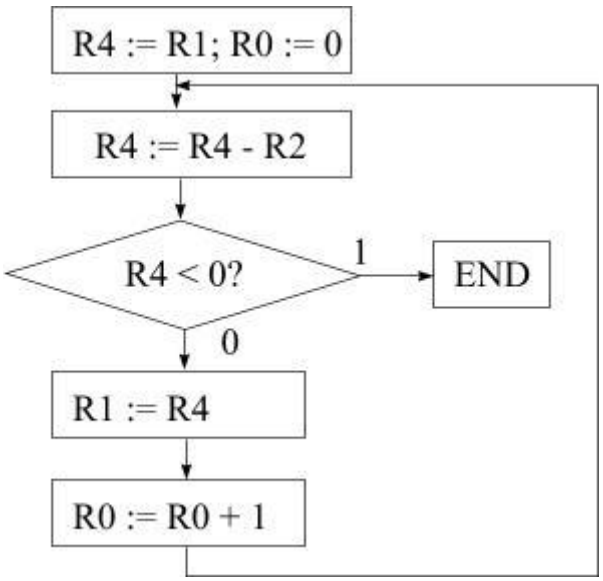

рис. 19 Алгоритм деления числа R1 на R2

Задача 7.1 Написать программу, реализующую вышеприведенный алгоритм.

Лабораторная работа 8.
Изучение приемов преобразования форматов чисел.

Рассмотрим задачу преобразования числа, имеющего размер в один байт, в двоично-десятичный (BCD) формат. Результатом преобразования будут три числа, имеющие размер в один байт, каждое из которых представляет один из разрядов (единицы, десятки и сотни) исходного числа.

Пример: R1 = 196, тогда результатом BCD-преобразования будут:

R7 (регистр единиц) = 6;

R6 (регистр десятков) = 9;

R3 (регистр сотен) = 1.

Алгоритм задачи представлен ниже.

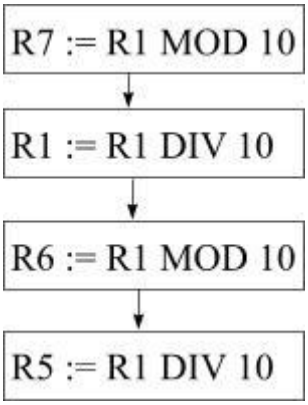

рис.20 Алгоритм преобразования числа в формат BCD

Задача 8.1 Написать программу, реализующую вышеприведенный алгоритм.

Лабораторная работа 9.
Работа с многобайтными числами.

В ходе лабораторной работы необходимо решить две задачи. При решении задач, необходимо разработать алгоритм для каждой задачи, описать постановку каждой задачи через описание используемых регистров и составить программу в терминах команд для микроконтроллера МК48.

Задача 9.1. Написать программу сложения 10 чисел, расположенных в резидентной памяти программ.

Задача 9.2. Написать программу сложения многобайтных чисел.

Задача 9.3. Написать программу вычитания многобайтных чисел.

Лабораторная работа 10.
Работа с портами ввода-вывода.

Приступая к работе, необходимо рассмотреть примеры в п.5, связанные с работой портов ввода-вывода. Обратите внимание, что в ряде случаев при работе с портами используются операции маскирования, которые требуют чтения данных только из определенных разрядов порта, или выдачу сигналов только в определенные разряды порта. Пример маскирования может быть представлен следующим образом:

```
IN A, P1
ANL A, #10011011B
```

При управлении различными периферийными устройствами, необходимо учитывать время этих сигналов установленных на разрядах порта. Для этого возможна реализация подпрограммы, работающей с таймером счетчиком МК48, или подпрограмма с пустыми циклами. Подпрограмма с пустыми циклами может быть представлена следующим образом:

```
DELAY:   MOV R2, #D      ; записываем число пустых циклов
COUNT:   DEC R2
         DJNZ R2, COUNT
         RET
```

При реализации подпрограммы использующей таймер-счетчик, необходимо учитывать, что эта функциональность реализуется через внутренне прерывание, которое вызывается событием переполнением таймера-счетчика, а обработка этого события организуется программным вектором таймера-счетчика. Запуск таймера может быть представлен следующим кодом:

```
MOV A, #(D/80-1)   ; вычисляем задержку
CRL A
MOV T, A           ; заполняем таймер
STRT T             ; запускаем таймер
EN TCNTI           ; разрешаем прерывание от таймера
```

Выдача сигнала в порт с задержкой сигнала может быть представлена следующим образом:

```
ON:    ANL P1, #00000001B
       CALL DELAY
OFF:   ORL P1, 11111110B
```

Задача 10.1 Написать программу реализующую выдачу сигналов в порт по диаграмме указанной на рисунке 21. Параметры t_1 и t_2 определяются по номеру варианта, $t_1 = N_{варианта} * 12мс$, $t_2 = 3с / N_{варианта}$. Полученные значения t_1 и t_2 округлить с точностью до 80 мкс.

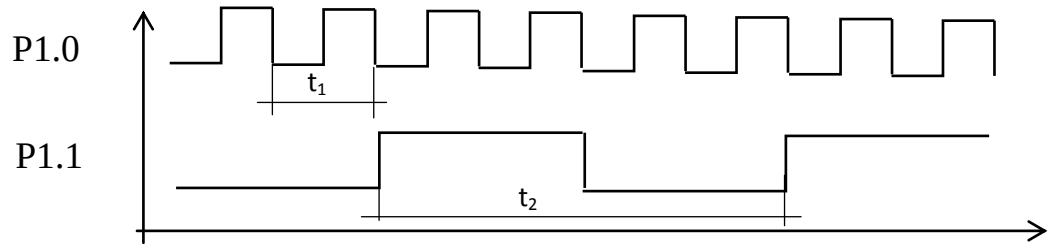

Рис. 21 Диаграмма сигналов на порт P1

Литература

1. Боборыкин А.В. Липовецкий Г.П., Литвинский Г.В. и др. Справочник. Однокристальные микроЭВМ. М.: МИКАП, 1994, – 400 с.
2. Угрюмов Е.П. Цифровая схемотехника. – СПб.: ПХВ-Петербург, 2001г. – 528с.
3. Водяхо А.И., Горнец Н.Н., Пузанков Д.Н. Высокопроизводительные системы обработки данных: Учебное пособие. – М.: Высшая школа, 1997.
4. Григорьев В.Л. Программирование однокристальных микропроцессоров.– М.: Энергоатомиздат,1987.
5. Сташин и др. Проектирование цифровых устройств на однокристальных микроконтроллерах/ В.В. Сташин, А.В. Урусов, О.Ф. Мологонцева. – М.:Энергоатомиздат, 1990. – 224 с.
6. Дмитриенко А.П., Старостенко О.В. Контроллер алфавитно-цифрового индикатора на базе однокристальной микроЭВМ// Микропроцессорные средства и системы. 1986. №6. С. 90-91.
7. Каган Б.М., Сташин В.В. Основы проектирования микропроцессорных устройств автоматики. М.: Энергоатомиздат,1987.
8. Клингман Э. Проектирование микропроцессорных систем. М.: Мир, 1980.
9. Балашов Е.П., Григорьев В.А., Петров Г.А. Микро- и мини-ЭВМ: Учебное пособие для вузов. – Л.: Энергоатомиздат, 1984.
10. Гук М.Ю. Аппаратные средства IBM PC: Энциклопедия. - СПб : Питер,1998. – 815 с.
11. Таненбаум Э. Архитектура компьютера – СПб : Питер, 2002. – 704 с.
12. Майоров С.А., Кириллов В.В., Приблуда А.А. Введение в микроЭВМ – Л.: Маштностроение -1988г.
13. Мальцев П.П., Долидзе Н., Критенко М Цифровые интегральные микросхемы. Справочник / Издательство: «Радио и связь»: 1994 г. – с. 240.