Введение

Создание микропроцессора (МП) явилось следствием развития и совершенствования технологии производства интегральных схем. Повышение степени интеграции микросхем привело к закономерному этапу в развитии вычислительной техники – реализации архитектуры ЭВМ на одной интегральной схеме.

Способность к программированию последовательности выполняемых функций, т.е. способность работать по заданной программе, является основным отличием МП от элементов «жесткой» логики (интегральных схем малой и средней степени интеграции). Аппаратные средства микропроцессора повторяют структуру процессора ЭВМ и включают: арифметико-логическое устройство, устройство управления, память. Микропроцессор может состоять из одной или нескольких интегральных схем.

Таким образом, микропроцессор – это программно-управляемое устройство, осуществляющее процесс обработки информации, управления им, построенное на одной или нескольких больших интегральных схемах (БИС).

За двадцать пять лет развития появилось несколько поколений микропроцессоров, отличающихся своими техническими характеристиками и инструментальными средствами программирования микропроцессорных систем. Один из этапов этого развития характеризуется появлением однокристальных микроЭВМ (ОЭВМ) с встроенными портами ввода/вывода и запоминающими устройствами. Особенности организации и программирования такой вычислительной системы (на примере ОЭВМ серии I8051 фирмы Intel) рассматриваются в данных указаниях.

Основными причинами широкого использования микропроцессорных систем (в том числе и ОЭВМ) в технике являются:

– использование в микропроцессорных системах цифрового способа представления информации, позволяющего значительно повысить помехоустойчивость создаваемых на их базе устройств, обеспечить простоту передачи и преобразования информации;

– применение программного способа обработки информации, позволяющего создавать в значительной степени унифицированные технические средства, отличающиеся практически лишь содержимым запоминающего устройства и специфическими устройствами ввода/вывода информации;

– компактность, высокая надежность и низкая потребляемая мощность микропроцессорных средств, обеспечивающих возможность расположения управляющих устройств, созданных на их основе, в непосредственной близости от управляемого оборудования, а иногда и встроенных в них;

– относительно низкая стоимость микропроцессорных средств.

Все эти факторы позволили МП в короткое время занять ведущее место в совершенствовании целых отраслей промышленности, создании гибких автоматизированных производств, систем передачи информации,

автоматизированных систем управления технологическими процессами, встраиваемых систем управления оборудованием и бытовыми приборами и т.д.

Микропроцессорная техника, как и любая другая область техники, имеет свою специфическую терминологию. Ниже приведены рекомендованные Международным центром научной и технической информации [1] термины и определения, используемые в данных методических указаниях.

Каждый раздел пособия завершается контрольными вопросами к разделу. Это сделано, в первую очередь, с целью дополнительного указания на наиболее существенные части изучаемого раздела, а также подготовить возможный перечень дополнительных вопросов, задаваемых при допуске и защите лабораторных работ, сдаче зачета и экзамена.

1. Основные термины и определения

Адрес – указание местоположения объекта в памяти ЭВМ.

Алгоритм – набор предписаний, однозначно определяющих содержание и последовательность выполнения операций для систематического решения определенной задачи.

Аналого-цифровой преобразователь (АЦП) – устройство, преобразующее непрерывный (аналоговый) сигнал в дискретные цифровые величины.

Арифметическо-логическое устройство (АЛУ) – функциональная часть процессора, выполняющая арифметические и логические действия над данными.

Ассемблер – системная обслуживающая программа, преобразующая символические инструкции в команды машинного языка и позволяющая производить диагностику, формирование ссылок для редактора связей и т.д.

Байт – обрабатываемый как единое целое элемент данных, состоящий из последовательности двоичных разрядов; в микроЭВМ обычно используется восьмибитовый байт.

Бит – один двоичный разряд машинного слова или единица информации, принимающая значения 0 или 1.

Буфер – запоминающее устройство для временного хранения данных с целью согласования асинхронно работающих устройств, либо область ОЗУ, временно резервируемая для выполнения процедуры ввода-вывода.

Встраиваемая микроЭВМ – микроЭВМ, конструктивно приспособленная для работы в составе приборов и оборудования.

Длина слова – количество битов в одном машинном слове.

Доступ (обращение) – процедура установления связи с запоминающим устройством для выборки/записи данных.

Ёмкость памяти – наибольший объем данных, выраженный в единицах информации, который может одновременно храниться в запоминающем устройстве.

Загрузчик – обслуживающая программа для загрузки объектной программы в ОЗУ.

Запоминающее устройство (ЗУ) – изделие, реализующее функциональную часть ЭВМ, которая предназначена для запоминания и (или) выдачи информации.

Интерпретатор – обслуживающая программа, осуществляющая пооператорную трансляцию и выполнение исходной программы.

Интерфейс – совокупность унифицированных технических и программных средств, необходимых для подключения внешних устройств к микропроцессорной системе или одной микропроцессорной системы к другой.

Канал передачи данных – совокупность технических средств, обеспечивающих передачу информации.

Команда – предписание, определяющее шаг процесса выполнения программы. Содержит указание операции, адрес операндов и другие служебные признаки.

Компилятор – обслуживающая программа, выполняющая трансляцию на машинный язык программы, записанной на исходном языке программирования.

Контроль четности – метод контроля данных, при котором сумма по модулю 2 двоичных единиц в машинном слове, включая контрольный разряд, должна иметь определенную четность, т.е. быть всегда четной или нечетной.

Косвенная адресация – система адресации, при которой адресная часть инструкции содержит адрес ячейки памяти, содержащей прямой адрес или другой косвенный адрес.

Магистраль – совокупность шин, связывающих собой все устройства микропроцессорной системы.

Машинное слово – последовательность битов или знаков, трактуемая в процессе обмена или обработки как единый элемент данных.

Микропроцессор (МП) – программно-управляемое устройство, осуществляющее процесс обработки цифровой информации и управления им, построенное, как правило, на одной или нескольких больших интегральных схемах.

Микропроцессорный комплект – совокупность микропроцессорных и других интегральных микросхем, совместимых по конструктивно-технологическому исполнению и предназначенную для совместного применения.

МикроЭВМ – ЭВМ, состоящая из микропроцессора, полупроводниковой памяти, средств связи с периферийными устройствами и при необходимости пульта управления и источника питания, объединенных общей конструкцией.

Монитор – записанная в ПЗУ системная программа, реализующая операции обмена с внешними устройствами и помогающая осуществить отладку программ.

Однокристальная микроЭВМ – микроЭВМ, выполненная в виде одной большой или сверхбольшой интегральной схемы.

Операнд – элемент данных, над которым выполняется операция.

Оперативное запоминающее устройство (ОЗУ) – ЗУ с прямой адресацией, отличающееся быстротой доступа.

Оператор – допустимая в языке программирования синтаксическая конструкция, отражающая определенное действие в программе (присвоение значения, передачу управления и т.д.).

Операционная система – комплекс взаимосвязанных управляющих и обслуживающих программ, обеспечивающих автоматическое управление вычислительными процессами и ресурсами ЭВМ при решении задач.

Параллельный порт – порт ввода-вывода, через который данные передаются и принимаются параллельно, т.е. одновременно все разряды, относящиеся к данному символу или блоку данных.

Подпрограмма – часть программы, допускающая многократное обращение к ней из различных точек программы.

Порт ввода-вывода – средство для подключения периферийных устройств к ЭВМ.

Последовательный порт – порт ввода-вывода, через который данные передаются и принимаются последовательно разряд за разрядом.

Постоянное запоминающее устройство (ПЗУ) – ЗУ с неизменяемым содержанием памяти.

Прерывание – временное прекращение выполнения текущей программы и переход к выполнению программы обслуживания устройства, вызвавшего прерывание.

Программа – последовательность инструкций, реализующих алгоритм. Программы обычно могут быть написаны: *а)* в двоичном или шестнадцатеричном (машинном) коде, который непосредственно воспринимается процессором; *б)* на языке типа Ассемблер; *в)* на языке высокого уровня.

Программа на исходном языке – программа, представленная в системе в исходном виде, т.е. написанная на одном из языков программирования. Требует для своего выполнения предварительного преобразования, например трансляции.

Программатор – специальное устройство для записи подготовленных пользователем программ в ППЗУ или РПЗУ.

Программная совместимость – возможность выполнения одних и тех же программ на ЭВМ различных типов с получением идентичных результатов.

Программное обеспечение – совокупность программ, обеспечивающих реализацию функций микроЭВМ, микропроцессорного устройства или системы.

Прямая адресация – система адресации, при которой адресная часть инструкции содержит адрес, определяющий непосредственно ячейку памяти или место на носителе, содержащее требуемый операнд.

Прямой доступ в память – метод, позволяющий с большой скоростью осуществлять загрузку данных с периферийного устройства прямо в оперативное ЗУ.

Регистр – функциональный блок для хранения машинного слова или его части.

Режим работы в реальном масштабе времени – режим работы системы, обеспечивающий прием к обработке данных по мере их поступления без каких-либо ограничений и выдачу результатов в требуемые интервалы времени, т.е. ход вычислительного процесса в системах реального времени задаётся обслуживаемыми устройствами.

Репрограммируемое постоянное запоминающее устройство (РПЗУ) – ЗУ, в котором информация, подлежащая хранению, заносится многократно.

Секционный микропроцессор – микропроцессор, полученный на основе соединения однотипных 2-, 4-, 8- или 16-разрядных микро-процессорных интегральных схем, каждая из которых имеет в своем составе АЛУ и несколько общих регистров. Параллельное соединение этих микросхем позволяет построить микроЭВМ с любой желаемой длиной машинного слова.

Сеть микроЭВМ – система соединенных между собой и обменивающихся информацией микроЭВМ.

Символ – отдельный знак из заданного набора условных обозначений, используемых для представления данных в ЭВМ.

Система команд – полный набор всех инструкций, допустимых в машинном языке данной ЭВМ.

Стек – оперативная память «магазинного» типа.

Счетчик команд – регистр, на основе содержимого которого вырабатывается адрес следующей команды.

Указатель стека – регистр, определяющий адрес верхней ячейки используемого стека.

Устройство ввода-вывода – устройство, обеспечивающее обмен данными между оперативной памятью ЭВМ и периферийными устройствами.

Цифро-аналоговый преобразователь (ЦАП) – устройство, преобра-зующее дискретный цифровой сигнал в непрерывный аналоговый сигнал.

Шина – группа линий передачи информации, объединенных общим функциональным признаком (например, шина данных, адресов, управления).

Эмуляция – имитация функционирования одной системы средствами другой системы без потери функциональных возможностей или искажения получаемых результатов.

Язык Ассемблер – символический язык программирования, структура операторов которого определяется форматами команд и данными машинного языка.

2. Архитектура микроконтроллеров серии MCS51

2.1.Общая характеристика микроконтроллеров

Микроконтроллеры (МК) MCS-51 являются функционально завершенными однокристальными микроЭВМ гарвардской архитектуры, содержащими все необходимые узлы для работы в автономном режиме, и предназначены для реализации различных цифровых алгоритмов управления.

Самым простым в функциональном плане и ранним по срокам разработки является кристалл I8051, который содержит минимальный базовый набор функциональных узлов, перечисленных ниже. Прочие микросхемы, являющиеся дальнейшим развитием кристалла I8051, отличаются от базовой версии улучшенной технологией изготовления, электрическими параметрами, алгоритмом программирования и наличием дополнительных аппаратно-реализованных возможностей. При этом сохраняются двоичная совместимость набора команд и интерфейс внешних устройств. Совместимым снизу-вверх является и набор доступных для программиста регистров и отдельных битов. Для большинства микросхем сохранено практически одинаковое расположение внешних выводов (рис.1). Назначение выводов приведено в табл. 1. Все микросхемы семейства обладают аналогичной архитектурой и имеют целый ряд общих узлов (рис.2):

– восьмиразрядный центральный процессор (ЦП), ориентированный на управление исполнительными устройствами. ЦП имеет встроенную схему 8-разрядного аппаратного умножения и деления чисел. Наличие в наборе команд большого количества операций для работы с прямоадресуемыми битами дает возможность говорить о процессоре для работы с битовыми данными (булевом процессоре);

– внутреннюю (расположенную на кристалле) память программ масочного или репрограммируемого типа, имеющую для различных кристаллов объем от 4 до 32 Кб (в некоторых версиях она отсутствует);

– не менее чем 128 байтное резидентное ОЗУ данных, которое используется для организации регистровых банков, стека и хранения пользовательских данных;

– не менее тридцати двух двунаправленных интерфейсных линий, организованных в четыре восьмиразрядных портов ввода/вывода, индивидуально настраиваемых на ввод или вывод информации;

– два 16-битных многорежимных счетчика/таймера, используемых для подсчета внешних событий, организации временных задержек и тактирования коммуникационного порта;

– двунаправленный дуплексный асинхронный приемопередатчик (UART), предназначенный для организации каналов связи между микроконтроллером и внешними устройствами с широким диапазоном скоростей передачи информации. Имеются средства для аппаратно-программного объединения микроконтроллеров в связанную систему;

– двухуровневую приоритетную систему прерываний, поддерживающую не менее 5 векторов прерываний от четырех внутренних и двух внешних источников событий;

– встроенный тактовый генератор.

Центральный процессор микроконтроллеров MCS-51 имеет следующие технические характеристики:

Разрядность АЛУ, бит 8

Число выполняемых команд 111

Длина команд, байт 1,2,3
Число регистров общего назначения (РОН) 32
Число прямоадресуемых битовых переменных 128
Число прямоадресуемых битов в области регистров
специального назначения 128
Максимальный объем памяти программ, Кб 64
Максимальный объем внешней памяти данных, Кб 64
Максимальный объем внутренней памяти данных, Кб 256
На частоте 12 МГц обеспечивается время
выполнения команд, мкс:

сложение 1
пересылки «регистр – внешняя память данных» . .2
умножение/деление 4

Методы адресации операндов:

–регистровый,
–косвенный,
–прямой,
–непосредственный.

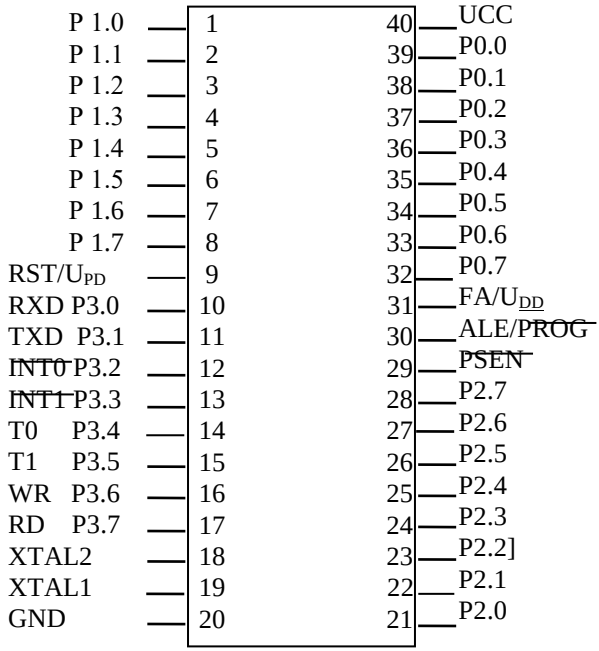


Рис.1. Расположение и обозначение выводов микросхемы I8051

Таблица 1

Назначение выводов микросхемы I8051

Обозначение	Номера выводов	Функция
1	2	3
P0.0 - P0.7	39-32	Порт 0. Восьмиразрядный двунаправленный порт ввода/вывода. Через порт осуществляется: • мультиплексная передача кода данных и младших разрядов кода адреса при обращении к внешней памяти; • ввод/вывод данных во время программирования и проверки памяти программ
P1.0 - P1.7	1-8	Порт 1. Восьмиразрядный квазидвунаправлен-ный порт ввода/вывода. Используется для передачи младших разрядов кода адреса при программировании и проверке ОЭВМ
P2.0 - P2.7	21-28	Порт 2. Восьмиразрядный квазидвунаправ-ленный порт ввода/вывода. Через порт осуществляется: • передача старших разрядов кода адреса при выборке из внешней памяти; • передача старших разрядов кода адреса и управляющих сигналов во время программирования и проверки после программирования
P3.0 - P3.7	10-17	Порт 3. Восьмиразрядный квазидвунаправлен-ный порт ввода/вывода. Через выводы порта 3 осуществляется передача сигналов управления: • RXD (P3.0). Прием последовательного кода (асинхронный режим) или ввод/вывод данных (синхронный режим); • TXD(P3.1). Передача последовательного кода (асинхронный режим) или вывод синхросигналов (синхронный режим); • INT0 (P3.2). Вход внешнего источника прерывания 0 или сигнал управления входом счетчика 0; • INT1 (P3.3). Вход внешнего источника прерывания 1 или сигнал управления входом счетчика 1; • T0 (P3.4). Вход счетчика событий 0; • T1 (P3.5). Вход счетчика событий 1;

Окончание табл.1

1	2	3
		• WR (P3.6). Сигнал управления записью. Строб фиксации байта данных, передаваемых из порта 0 во внешнюю память данных; • RD (P3.7). Строб управления считыванием данных из внешней памяти данных через порт 0
ALE/$\overline{PROG}$	30	Выходной сигнал $\overline{ALE}$ используется для фиксации адреса внешней памяти. Во время программирования СППЗУ на выводе 30 подается импульс программирования $\overline{PROG}$
$\overline{PSEN}$	29	Управляющий сигнал, по которому данные из внешней памяти программ поступают на шину данных во время операций выборки
RST\U_{PD}	9	Сброс/аварийный источник питания
GNP	20	Потенциал земли
EA/U_{DD}	31	При подаче на вход EA сигнала высокого уровня напряжения (ТТЛ) выполняются команды из встроенного ПЗУ (СППЗУ), если адрес, содержащийся в PC, меньше, чем 4096. При подаче на вход EA сигнала низкого уровня (ТТЛ) будет осуществляться выборка по всем адресам (0–64Kb) из внешней памяти программ. На контактах EA/U_{DD} подается напряжение +21В при программировании СППЗУ
XTAL1	19	Вход усилителя-генератора синхросигнала с высоким коэффициентом усиления, заземляется при использовании внешнего задающего генератора. Выход усилителя-генератора синхросигнала
XTAL2	18	Вход для встроенного задающего генератора. Прием сигналов внешнего задающего генератора. Напряжение питания +5В во время работы, программирования и проверки

2.2. Структурная организация МК

Основу структурной схемы МК составляет внутреннее двунаправленная 8-битная шина, которая связывает между собой все узлы и устройства: резидентную память, АЛУ, блок регистров специальных функций, устройство управления и порты ввода/вывода.

Рассмотрим основные элементы структуры и особенности организации вычислительного процесса в МК.

Арифметико-логическое устройство. Восьмибитное АЛУ может выполнять арифметические операции сложения, вычитания, умножения и

деления; логические операции И, ИЛИ, исключающее ИЛИ, а также операции циклического сдвига, сброса, инвертирования и т. п. В АЛУ имеются программно недоступные регистры Т1 и Т2, предназначенные для временного хранения операндов, схема десятичной коррекции и схема формирования признаков.

Простейшая операция сложения используется в АЛУ для инкрементирования содержимого регистров, продвижения регистра-указателя данных и автоматического вычисления следующего адреса. Простейшая операция вычитания используется в АЛУ для декрементирования регистров и сравнения переменных.

Простейшие операции автоматически образуют «тандемы» для выполнения в АЛУ таких операций, как, например, инкрементирование 16-битных регистровых пар. В АЛУ реализуется механизм каскадного выполнения простейших операций для реализации сложных команд. Так, например, при выполнении одной из команд условной передачи управления по результату сравнения в АЛУ трижды инкрементируется счетчик команд, дважды производится чтение из памяти, выполняется арифметическое сравнение двух переменных, формируется 16-битный адрес перехода и принимается решение о том, делать или не делать переход по программе. Все перечисленные операции выполняются в АЛУ всего лишь за две микросекунды.

Важной особенностью АЛУ является его способность оперировать не только байтами, но и битами. Отдельно программно доступные биты могут быть установлены, сброшены, инвертированы, переданы, проверены и использованы в логических операциях. Для управления объектами часто применяются алгоритмы, содержащие операции над входными и выходными булевскими переменными (истина/ложь), реализация которых средствами обычных микропроцессоров сопряжена с определенными трудностями.

Таким образом, АЛУ может оперировать четырьмя типами информационных объектов: булевскими (1 бит), полубайтовыми (4 бита), байтами (8 бит) и адресными (16 бит). В АЛУ выполняется 51 различная операция пересылки или преобразования этих данных. Так как используется 11 режимов адресации (7 для данных и 4 для адресов), то путем комбинирования «операция/режим адресации» базовое число команд 111 расширяется до 255 из 256 возможных при однобайтном коде операции.

Резидентная память. Память программ и память данных, разрешенные на кристалле I8051, физически и логически разделены, имеют различные механизмы адресации, работают под управлением разных сигналов и выполняют различные функции.

Память программ (ПП) предназначена для хранения команд, констант, управляющих слов инициализации, таблиц перекодировки, входных и выходных переменных и т. п. ПП имеет 16-битную шину адреса, через которую обеспечивается доступ из счетчика команд или из регистра-указателя данных. Последний выполняет функции базового регистра при косвенных переходах по программе или используется в командах, оперирующих с таблицами.

Память данных (ПД) предназначена для хранения переменных в процессе выполнения прикладной программы, адресуется одним байтом и имеет ёмкость 128 байт. Кроме того, к адресному пространству ПД примыкают адреса регистров специальных функций (РСФ), которые перечислены в **т**абл.2. Память программ так же как и память данных, может быть расширена до 64 Кбайт путем подключения к внешней БИС.

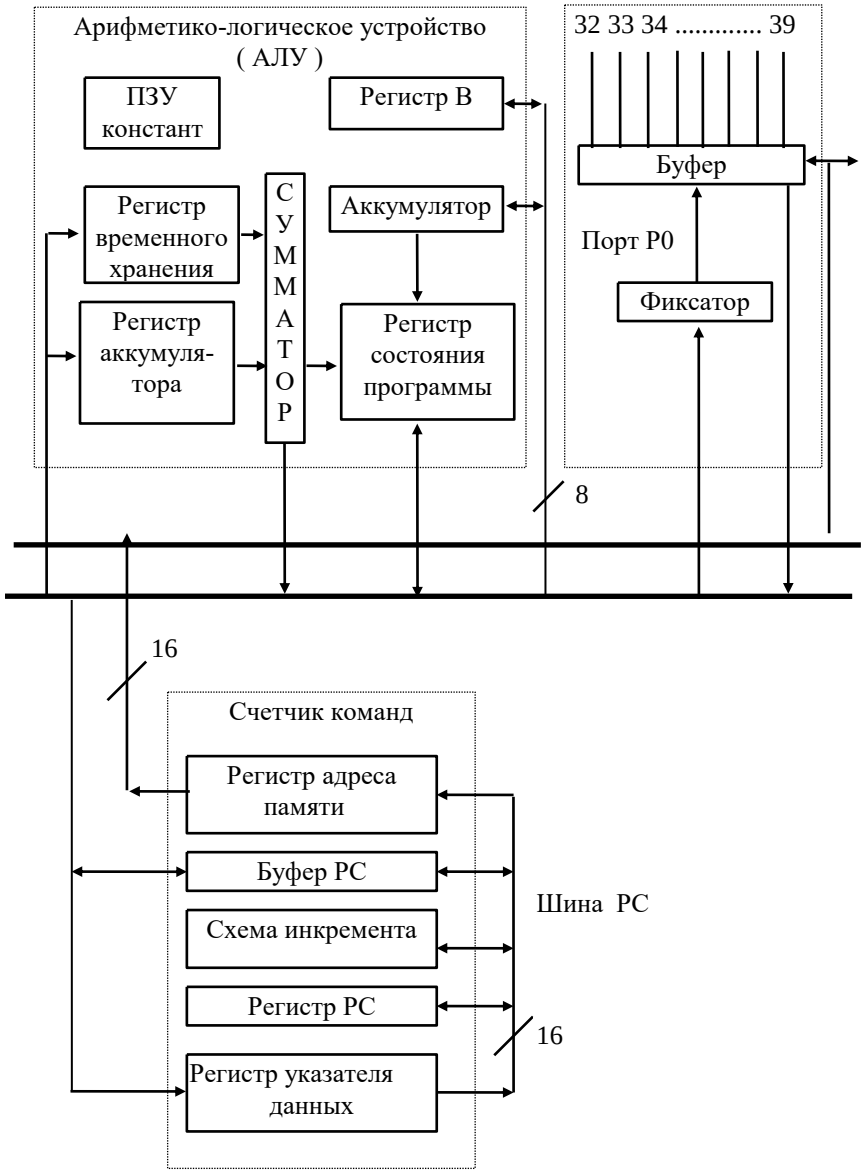


Рис. 2. Структурная схема ОМЭВМ

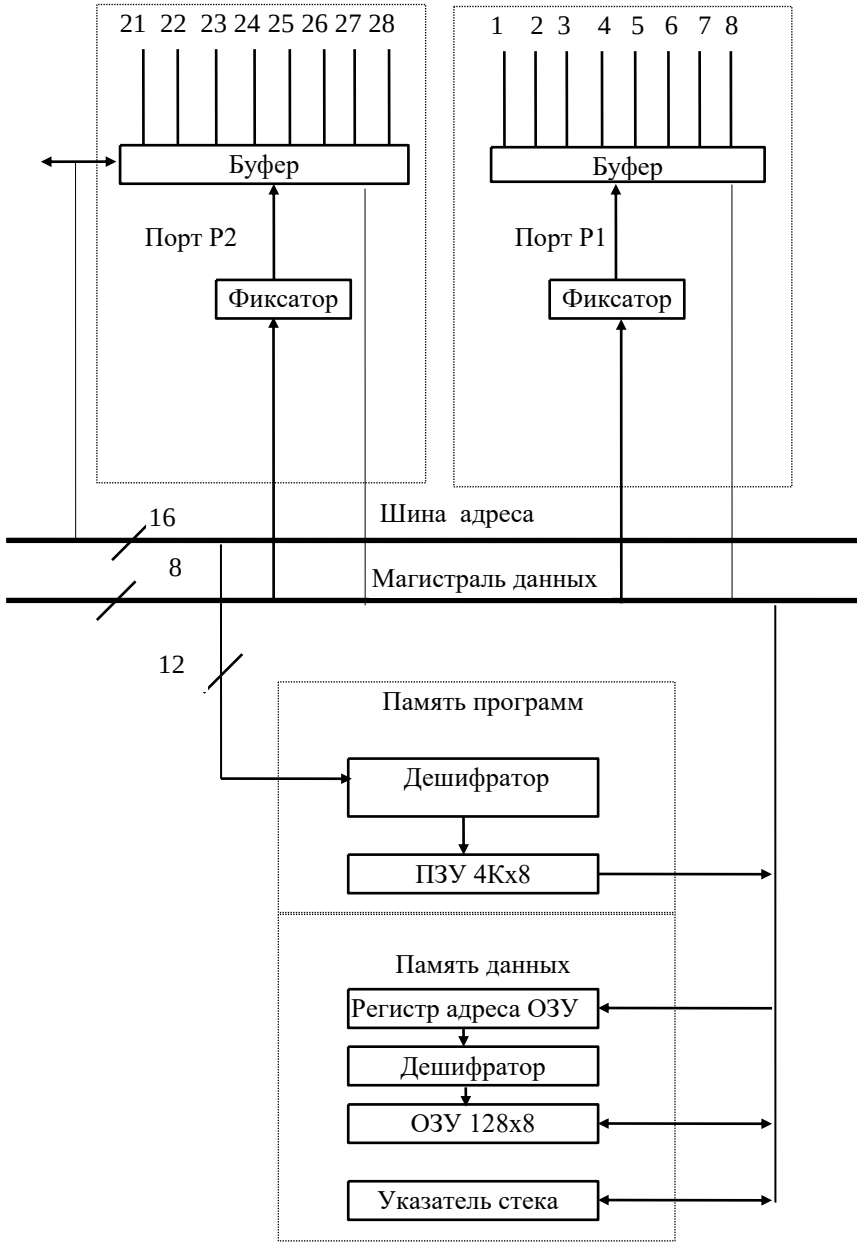


Рис.2. Продолжение

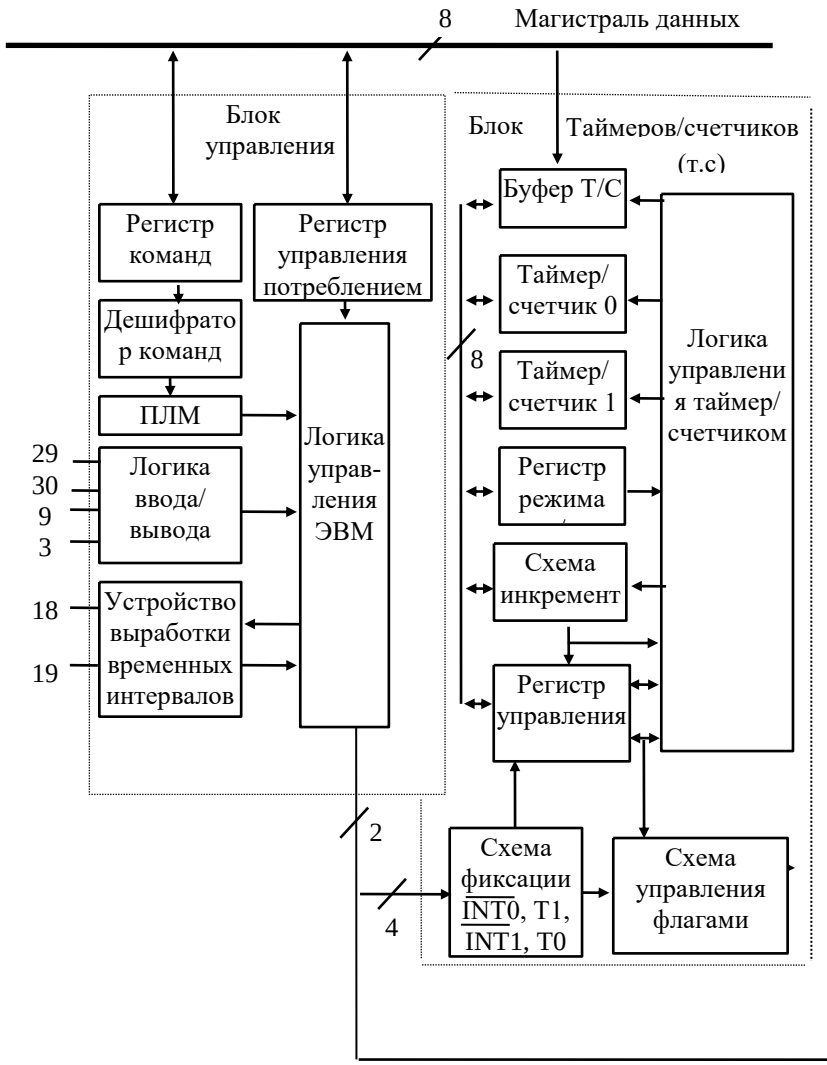


Рис.2. Продолжение

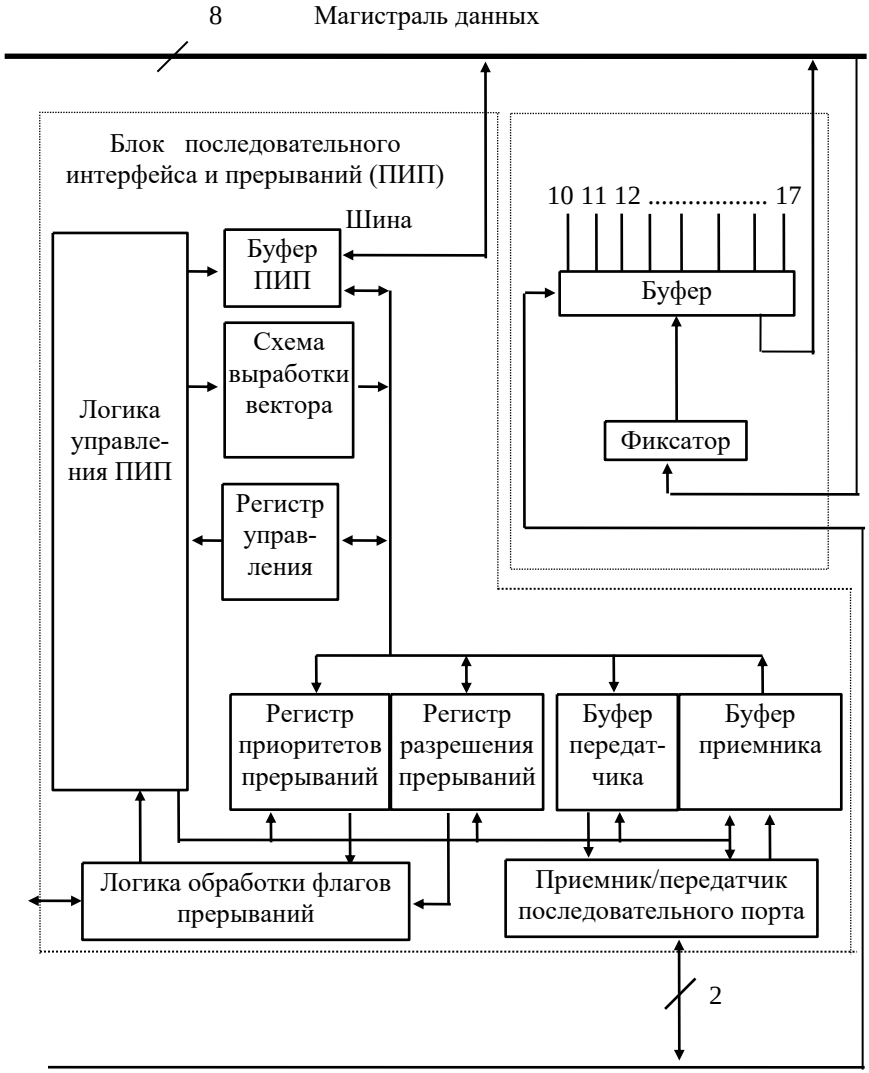


Рис.2. Окончание

Таблица 2

Блок регистров специальных функций

Обозначение	Назначение	Адрес
ACC *	Аккумулятор	0E0h
B *	Регистр-расширитель аккумулятора	0F0h
PSW *	Слово состояния программы	0D0h
SP	Регистр-указатель стека	81h
DPTR	» данных (DPH)	83h
	» (DPL)	82h
P0 *	Порт 0	80h
P1 *	» 1	90h
P2 *	» 2	0A0h
P3 *	» 3	0B0h
IP *	Регистр приоритетов прерываний	0B8h
IE *	» разрешения прерываний	0A8h
TMOD	» режимов таймеров/счётчиков	89h
TCON *	» управления таймерами/счётчиками	88h
TH0	Таймер/счётчик 0. Старший байт	8Ch
TL0	» Младший байт	8Ah
TH1	Таймер/счётчик 1. Старший байт	8Dh
TL1	» Младший байт	8Bh
SCON *	Регистр управления приёмопередатчиком	98h
SBUF	Буфер приёмопередатчика	99h

Примечание. Регистры, имена которых отмечены знаком (*), допускают адресацию отдельных бит.

При выполнении многих команд в АЛУ формируется ряд признаков операции (флагов), которые фиксируются в регистре ССП. В табл. 3 приводится перечень флагов ССП, даются их символические имена и описываются условия их формирования.

Таблица 3

Формат слова состояния программы (ССП)

Обозначение	Позиция	Назначение
C	PSW.7	Флаг переноса. Устанавливается и сбрасывается аппаратными средствами или программой при выполнении арифметических и логических операций
AC	PSW.6	Флаг вспомогательного переноса. Устанавливается и сбрасывается только аппаратными средствами при выполнении команд сложения и вычитания и сигнализирует о переносе или заеме в бите 3
F0	PSW.5	Флаг 0. Может быть установлен, сброшен или проверен программой как флаг, специфицируемый пользователем
RS1	PSW.4	Выбор банка регистров. Устанавливается и сбрасывается программой для выбора рабочего банка регистра (см. примечание)
RS0	PSW.3	
OV	PSW.2	Флаг переполнения. Устанавливается и сбрасывается аппаратно при выполнении арифметических операций.
-	PSW.1	Не используется.
P	PSW.0	Флаг паритета. Устанавливается и сбрасывается аппаратно в каждом цикле команды и фиксирует нечетное/четное число единичных бит в аккумуляторе, т.е. выполняет контроль по четности.
Примечание.		Выбор рабочего банка регистров:

RS1	RS0	Банк	Границы адресов
0	0	0	00H-07H
0	1	1	08H-0FH
1	0	2	10H-17H
1	1	3	8H-1FH

АЛУ не управляет флагами селекции банка регистров (RS0, RS1) и их значение полностью определяется прикладной программой и используется для выбора одного из четырех регистровых банков.

Процессор в МК имеет в своей основе аккумулятор, однако он может выполнять множество команд и без участия аккумулятора. Например, данные могут быть переданы из любой ячейки ПД в любой регистр, любой регистр

может быть загружен непосредственным операндом и т.д. Многие логические операции могут быть выполнены без участия аккумулятора. Кроме того, переменные могут быть инкрементированы, декрементированы и проверены (test) без использования аккумулятора. Флаги и управляющие биты могут быть проверены и изменены аналогично.

Регистры-указатели. Указатель стека (УС) может адресовать любую область ПД. Его содержимое инкрементируется прежде чем данные будут запомнены в стеке в ходе выполнения команд PUSH и CALL. Содержимое УС декрементируется после выполнение POP и RET. В процессе инициализации МК после сигнала Сброс в УС автоматически загружается код 07h. Это означает, что если прикладная программа не переопределяет стек, то первый элемент данных в стеке будет располагаться в ячейки ПД с адресом 08h.

Двухбайтный регистр-указатель данных (УД) обычно используется для фиксации 16-битного адреса в операциях с обращением к внешней памяти. Командами МК регистр-указатель данных может быть использован или как 16-битный регистр, или как 2 независимых 8-битных регистра (DPH и DPL).

Таймер/счетчик. В составе средств МК51 имеются регистровые пары с символическими именами TH0, TL0 и TH1, TL1, на основе которых функционируют два независимых программно-управляемых 16-битных таймера счетчика событий.

Буфер последовательного порта. Регистр с символическим именем SBUF представляет собой два независимых регистра: буфер приемника и буфер передатчика. Загрузка байта в SBUF немедленно вызывает начало процесса передачи через последовательный порт. Когда байт считывается из SBUF, это значит, что его источником является приемник последовательного порта.

Регистры специальных функций. Регистры с символическими именами IP, IE, TMOD, TCON, SCON используются для фиксации и программного изменения управляющих бит и бит состояния схемы прерывания, таймера/счетчика, приемопередатчика последовательного порта МК. Их организация будет описана ниже при рассмотрении особенности работы МК в различных режимах.

Порты ввода/вывода информации. Все четыре порта МК предназначены для ввода и вывода информации побайтно. Каждый порт содержит управляемые регистр-защелку, входной и выходной буферы.

Выходные буферы портов 0 и 2, а также входной буфер порта 0 используется при обращении к внешней памяти (ВП). При этом через порт 0 в режиме временного мультиплексирования сначала выводится младший байт адреса ВП, затем выдается или принимается байт данных. Через порт 2 выводится старший байт адреса в тех случаях, когда разрядность адреса равна 16 бит.

Все выводы порта 3 могут быть использованы для реализации альтернативных функций, подробно рассмотренных в соответствующем разделе.

Порт 0 является двунаправленным, а порты 1, 2 и 3 квази-двунаправленными. Каждая линия портов может быть использована независимо для ввода и вывода информации.

По сигналу «Сброс» в регистры-защелки всех портов автоматически записываются единицы, настраивающие их тем самым на режим ввода.

Все порты могут быть использованы для организации ввода/вывода информации по двунаправленным линиям передачи. Однако порты 0 и 2 не могут быть использованы для этой цели в случае, если МК-система имеет внешнюю память, связь с которой реализуется через общую разделяемую шину адреса/данных, работающую в режиме временного мультиплексирования.

Контрольные вопросы

1. Дайте характеристику микроконтроллеров серии MCS-51.
2. Сформулируйте назначение и дайте краткую характеристику основных внешних выводов ОЭВМ I8051.
3. Назовите и дайте характеристику основным структурным элементам ОЭВМ.
4. Дайте характеристику регистровой структуры ОЭВМ.
5. Дайте характеристику разрядов регистра слова состояния программы.

3. Организация памяти в микроконтроллерах серии MCS-51. Представление программ и данных в памяти МК

3.1.Разделение памяти программ и данных

Как уже было сказано выше, вся серия MCS-51 имеет ***гарвардскую*** архитектуру – раздельное адресное пространство программ и данных (рис.3). Такое разделение позволяет осуществлять доступ к памяти данных по 8-битным адресам, разрядность которых позволяет осуществить их быструю обработку ЦП. Тем не менее, также возможно использование 16-битных адресов данных с помощью регистра указателя данных **DPTR**.

Память программ доступна исключительно для чтения (если не применять ее искусственное объединение с памятью данных путем использования внешних цепей). Объем адресуемого пространства может составлять до 64К, хотя на различных кристаллах, в составе которых имеется резидентное масочное ПЗУ (**ROM**), УФРПЗУ (**EPROM**) или ППЗУ (**OTP-ROM**) содержится лишь 4, 8, 16 или 32 Кбайт, в зависимости от типа микросхемы, причем при обращении к этим областям используются циклы обмена по внутренней магистрали. При необходимости пользователь может расширять память программ установкой внешнего ПЗУ. При считывании команд из внешней программной памяти формируется инверсный сигнал **PSEN** (Program Store Enable) для всех адресов, кроме области встроенного ПЗУ.

Память данных отделена от памяти программ. В этой области возможна адресация 64 Кбайт внешнего ОЗУ. ЦП генерирует инверсные сигналы чтения **RD** и записи **WR**, необходимые для обращения к внешней памяти данных.

Внешняя память программ и внешняя память данных могут комбинироваться путем совмещения сигналов **RD** и **PSEN** по схеме «логического И» для получения строба внешней памяти (программ/ данных).

3.2.Память программ

Карта нижней области памяти программ изображена на рис.4. После сигнала сброса микроконтроллер начинает выполнение программы с адреса **0000h**. Из рис.4 видно, что каждому прерыванию соответствует свой фиксированный адрес в нижней области памяти программ. Прерыва-ние заставляет процессор осуществлять переход по этому адресу, с которого начинается выполнение подпрограммы обработки прерывания. «Внешнее прерывание 0» EINT0 (вывод **INT0**) имеет адрес **0003h**. Если прерывание не используется, этот адрес доступен как и остальные.

Адреса прерываний расположены с интервалом в 8 байт:

* 0003h - External interrupt 0 (внешнее прерывание 0 - вывод INT0);
* 000Bh - Timer 0 (прерывание от 1-го таймера);
* 0013h - External interrupt 1 (внешнее прерывание 1 - вывод INT1);
* 001Bh - Timer 1 (прерывание от 2-го таймера).

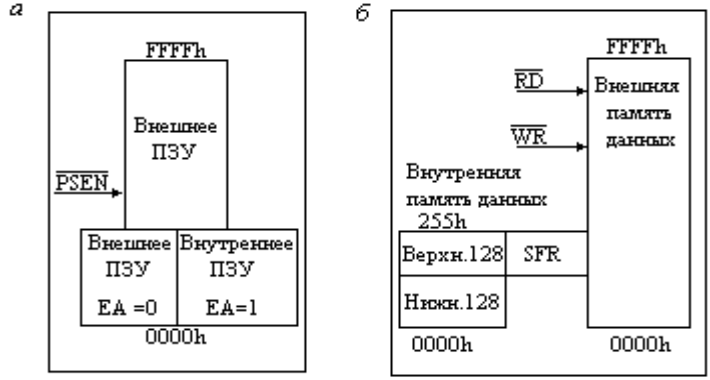


Рис.3. Организация памяти в микроконтроллерах серии MCS-5:
а – память программ; б – память данных (чтение/запись)

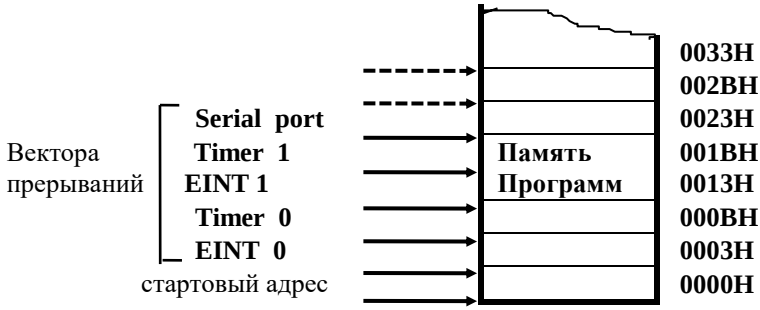


Рис.4. Память программ

Если обработчик прерывания имеет небольшой размер, то он может полностью разместиться в этих восьми байтах. Более длинная подпрограмма , используя команду перехода, может быть размещена в любом другом месте памяти, не затрагивая другие векторы прерываний (если те используются). Например:

```
INT0_Vector:    JMP     INT0_Start
                . . .
Timer0_Vector:
                . . .
                RETI
INT0_Start:
                . . .
                RETI
```

Младшие 4 Кбайт (8, 16, 32 Кбайт – для различных модификаций) памяти программ размещаются на кристалле или находятся во внешнем ПЗУ.

В микросхеме с 4К ПЗУ при подключении **EA** к **UCC** адреса с 0000h, по 0FFFh располагаются во внутреннем ПЗУ, а адреса с 1000h по FFFFh – во внешнем.

В 8К-байтном кристалле, если **EA= UCC**, то адреса 0000h – 1FFFh относятся к внутреннему ПЗУ, а 2000h – FFFFh – к внешнему. Для 16К-байтного кристалла эти адреса – 0000h-3FFFh и 4000h – FFFFh соответственно.

Если вывод **EA** подключен к **GND** (земле), то вся программа располагается во внешнем ПЗУ. Кристаллы без ПЗУ (**ROMless**) для правильной работы должны иметь этот вывод постоянно подключенным к **GND**.

Строб чтения внешнего ПЗУ – **PSEN** используется во всех обращениях к внешней памяти программ и является неактивным во время обращения к памяти , расположенной на кристалле.

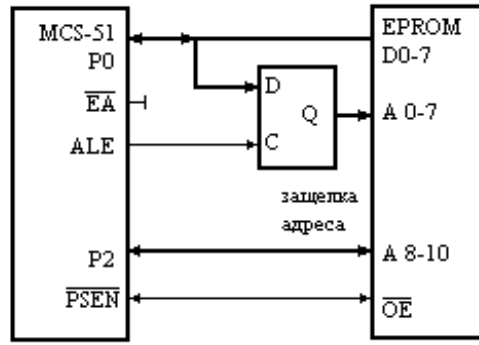


Рис.5. Использование внешней памяти программ

Аппаратная реализация внешней памяти программ показана на рис.5**.** Отметим, что 16 линий ввода-вывода портов 0 и 2 используются для обращения к внешней памяти программ. Порт 0 (**P0**) является мультиплексированной шиной адреса/данных. На него выдается младший байт

программного счетчика (**PCL**), после чего он переводиться в третье состояние для чтения байта из памяти программ. В период установки на линиях **P0** младшего байта программного счетчика, сигнал **ALE** фиксирует его на внешней защелке адреса. Одновременно в порт 2 выдается старший байт программного счетчика (**PCH**). Байт данных выдается из ПЗУ в микроконтроллер по стробу сигнала **PSEN**.

3.3. Память данных

В правой части рис.3 показано расположение внутренней и внешней памяти данных, доступной пользователю. Подключение внешней памяти данных объемом до 2 Кбайт показано на рис.6, но при этом считывание программы производится из внутреннего ПЗУ. Порт 0 используется для связи с внешним ОЗУ мультиплексированной шиной адреса/данных, а 3 линии порта **P2** предназначены для выбора страницы ОЗУ. Для управления внешней памятью процессор генерирует сигналы **RD** и **WR**.

При этом возможна адресация до 64 Кбайт внешней памяти данных. Адрес может иметь размер в 1 или 2 байта. Однобайтная адресация часто используется при страничной организации ОЗУ. Выбор страниц осуществляется линиями портов ввода-вывода, как показано на рис.6. При использовании двухбайтной адресации старший байт адреса выводится через порт **P2**.

Внутренняя память данных изображена на рис.7. Она может быть разделена на 3 условных блока: «**нижний**» (lower), «**верхний**» (upper) и пространство **Регистров Специальных Функций** (SFR**).**

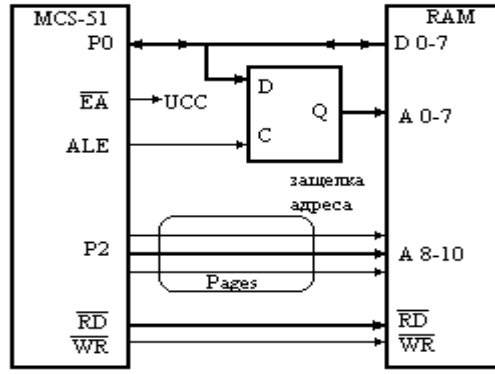


Рис.6. Использование внешней памяти данных

FFh	**Область внутреннего ОЗУ**	**Область регистров специ-**
. . .	**«Upper 128»**	**альных функций (SFR)**
. . .	Доступна в режиме косвенной	Доступна в режиме прямой
80h	адресации (на I8051-отсутствует)	адресации

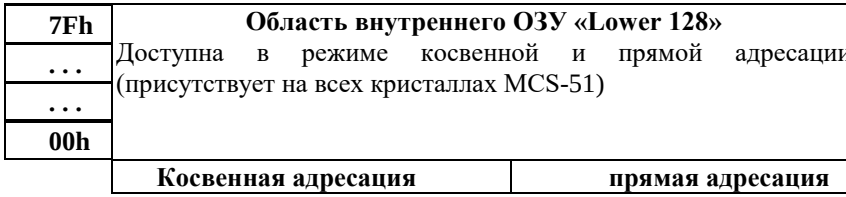


Рис.7. Внутренняя память данных

Внутренняя память данных имеет байтовую адресацию, которая подразумевает 256 байт адресного пространства. Нижние 128 байт ОЗУ присутствуют на всех кристаллах MCS-51 и показаны на рис.8. Первые 32 байта сгруппированы в 4 банка по 8 регистров. Инструкции программы могут оперировать с ними как с регистрами общего назначения **R0**-**R7**. Два бита **ССП** (P**SW**) определяют, какой из банков используется в текущий момент. Это позволяет более эффективно использовать память программ, поскольку регистровые инструкции выполняются быстрее. Переключение регистровых банков эффективно заменяет операция сохранения регистров в стеке на время обработки процедуры прерывания.

Следующие за регистровыми банками 16 байт образуют блок побитно адресуемого пространства. Набор инструкций семейства MCS-51 содержит широкий выбор операций над битами, а 128 бит (16 байт 8) в этом блоке могут быть прямо адресованы. Битовые адреса имеют значения от 00h до 7Fh.

Все байты в нижней 128-байтной половине памяти могут адресоваться как прямо, так и косвенно. Верхняя 128-байтная половина ОЗУ в микро-схеме I8051 отсутствует, но имеется в кристаллах с 256 байтами ОЗУ.

В табл. 2 показано размещение регистров специальных функций (**SFR**). Эти регистры включают в себя регистры портов, таймеры, средства управления периферией и т.д. Они доступны только при прямой адресации. Для 16 адресов в пространстве SFR имеется возможность как битовой, так и байтовой адресации (табл.4). Побитно адресуемыми регистрами являются регистры, шестнадцатиричные адреса которых заканчивается на 0 или 8. Битовые адреса в этой области располагаются со значения 80h по FFh.

Вся серия имеет базовый набор SFR, как в микросхеме I8051 по тем же адресам. Однако в кристаллах, представляющих собой дальнейшее развитие I8051, имеются и некоторые дополнительные регистры, отсутствующие в базовом варианте. Этим микросхемам свойственны свои уникальные функциональные особенности (такие, как наличие матрицы программируемых счетчиков - **PCA**, сторожевого таймера - **HWDT**, АЦП, улучшенная система прерываний и т.д.). Это подразумевает наличие новых регистров, имеющих собственные адреса в пространстве SFR; реально использующих некоторые битовые поля, зарезервированные для битовой адресации SFR в микроконтроллере I8051.

7Fh . . . 30h	Побайтно адресуемая область ОЗУ							
	Побитно адресуемая область ОЗУ (битовые адреса 00h-7Fh)							
2Fh	7Fh	7Eh	7Dh	7Ch	7Bh	7Ah	79h	78h
2Eh	77h	76h	75h	74h	73h	72h	71h	70h
2Dh	6Fh	6Eh	6Dh	6Ch	6Bh	6Ah	69h	68h
2Ch	67h	66h	65h	64h	63h	62h	61h	60h
2Bh	5Fh	5Eh	5Dh	5Ch	5Bh	5Ah	59h	58h
2Ah	57h	56h	55h	54h	53h	52h	51h	50h
29h	4Fh	4Eh	4Dh	4Ch	4Bh	4Ah	49h	48h
28h	47h	46h	45h	44h	43h	42h	41h	40h
27h	3Fh	3Eh	3Dh	3Ch	3Bh	3Ah	39h	38h
26h	37h	36h	35h	34h	33h	32h	31h	30h
25h	2Fh	2Eh	2Dh	2Ch	2Bh	2Ah	29h	28h
24h	27h	26h	25h	24h	23h	22h	21h	20h
23h	1Fh	1Eh	1Dh	1Ch	1Bh	1Ah	19h	18h
22h	17h	16h	15h	14h	13h	12h	11h	10h
21h	0Fh	0Eh	0Dh	0Ch	0Bh	0Ah	09h	08h
20h	07h	06h	05h	04h	03h	02h	01h	00h

1Fh . . . 18h	Регистровый банк 3
17h . . . 10h	Регистровый банк 2
0Fh . . . 08h	Регистровый банк 1
07h . . . 00h	< — SP после RESET Регистровый банк 0

Рис.8. Нижние 128 байт внутреннего ОЗУ

При программировании доступны следующие SFR-регистры:

Аккумулятор (адрес 0E0h). Кроме обращения к аккумулятору командами, использующих мнемонику «**A**», имеется возможность побито-вой или побайтной адресации, как SFR-регистра. Команды, предназначенные для побитной работы с аккумулятором, применяют мнемонику «**ACC**». Для

обращения к определенному биту аккумулятора на языке ассемблера MCS-51 достаточно записать: **ACC.N**, где N – номер требуемого бита:

MOV ACC.0, C; пересылка в младший бит аккумулятора флага переноса.

Регистр B (адрес 0F0h). Используется при операциях умножения и деления, а также как сверхоперативный регистр. Обращение к нему, как к регистру SFR , производится аналогично аккумулятору.

Слово Состояния Программы (PSW) (адрес 0D0h). Этот регистр микроконтроллера содержит информацию о состоянии программы. Кроме неявного его использования битовыми и арифметическими командами, может быть адресован как регистр SFR, в том числе и побитно. Каждый бит PSW имеет свою собственную мнемонику, например: «PSW.AC».

Указатель стека SP (адрес 081h). Используется для указания на вершину стека в операциях записи в стек и чтении из него. Неявно используется такими командами, как PUSH, CALL, RET, RETI, POP. По аппаратному сбросу ЦП устанавливается в значение 07h (область стека в этом случае соответственно начинается с адреса внутренней памяти данных 08h) и инкрементируется при каждой записи в стек. Запись в SFR-регистр указателя стека производится для предопределения положения стека во внутренней памяти данных микроконтроллера. Здесь возможна только байтная адресация.

Указатель данных DPTR (адреса 082h, 83h). Состоит из двух байт: старшего DPH и младшего DPL. Используя как 16 или 8-битовый указатель адреса при обращении к внешней памяти или выполнения команды перехода по косвенному адресу. Обращение – только байтное, отдельно к старшему и младшему байту.

Порты P0-P3 (адреса 80h, 90h, 0A0h, 0B0h). Этим адресам соответствуют регистры-защелки четырех 8-битных портов ввода-вывода микроконтроллера P0, P1, P2 и P3.

Буфер последовательного порта SBUF (адрес 99h). SBUF представляет собой два отдельных регистра – «буфер приемника» – при чтении из порта и «буфер передатчик» – при записи в порт. Доступен только для байтной адресации.

Регистр управления последовательным портом SCON(адрес 098h). Предназначен для управления работой последовательного порта. Обращение к данному регистру может быть как побайтным, так и побитным.

Регистры таймера/счетчика TL0, TL1, TH0, TH1 (адреса 8Ah, 8Bh, 8Ch, 8Dh). Образуют 16-битные регистры таймеров/счетчиков 0 и 1. Подробно описаны в разделе «Организация таймеров/счетчиков». Обращение к регистрам – только байтное.

Регистр режима таймера/счетчика TMOD (адрес 89h).

Регистр управления таймера/счетчика TCON (адрес 88h). Предназначены для управления работой таймера/счетчика МК.

Регистр разрешения прерываний IE (адрес 0A8h).

Регистр приоритетов прерываний IP (адрес 0B8h). Определяют работу системы прерываний микроконтроллера. Возможно как побитное, так и побайтное обращение к регистрам.

Регистр управления энергопотреблением PCON (адрес 87h). Содержит бит управления скоростью работы последовательного канала и биты, управляющие энергопотреблением в CHMOS-кристаллах (I8051 и т.п.).

Таблица 4

Регистры специальных функций при побитной адресации

	Мнемоника регистра / битовые адреса								
Адрес	7	6	5	4	3	2	1	0	Имя
F0h	— F7h	— F6h	— F5h	— F4h	— F3h	— F2h	— F1h	— F0h	B
	—	—	—	—	—	—	—	—	
D0h	CV D7h	AC D6h	F0 D5h	RS1 D4h	RS0 D3h	OV D2h	— D1h	P D0h	PSW
B8h	— —	— —	PT2 BDh	PS BCh	PT1 BBh	PX1 BAh	PT0 B9h	PX0 B8h	IP
B0h	— B7h	— B6h	— B5h	— B4h	— B3h	— B2h	— B1h	— B0h	P3
A8h	EA AFh	— —	ET2 ADh	ES ACh	ET1 ABh	EX1 AAh	ET0 A9h	EX0 A8h	IE
A0h	— A7h	— A6h	— A5h	— A4	— A3h	— A2h	— A1h	— A0h	P2
98h	SM0 9Fh	SM1 9Eh	SM2 9Dh	REN 9Ch	TB8 9Bh	RB8 9Ah	TI 99h	RI 98h	SCON
90h	— 97h	— 96h	— 95h	— 94h	— 93h	— 92h	— 91h	— 90h	P1
88h	TF1 8Fh	TR1 8Eh	TF0 8Dh	TR0 8Ch	IE1 8Bh	IT1 8Ah	IE0 89h	IT0 88h	TCON
80h	— 87h	— 86h	— 85h	— 84h	— 83h	— 82h	— 81h	— 80h	P0

Контрольные вопросы

1. Дайте общую характеристику организации памяти в ОЭВМ.
2. Дайте характеристику памяти программ ОЭВМ.
3. Дайте характеристику памяти данных ОЭВМ.
4. Охарактеризуйте особенности использования внешней памяти программ и данных (с точки зрения аппаратной реализации).
5. Охарактеризуйте возможности и ограничения в использовании внутренней памяти данных ОЭВМ.

4. Организация и программирование портов ввода/вывода

4.1. Параллельные порты ввода / вывода

В микроконтроллерах MCS-51 существуют четыре многофункциональных 8-битовых порта ввода/вывода: **P0**, **P1**, **P2** и **P3**, предназначенные для обмена информацией с различными внешними устройствами, такими, как внешняя память программ и данных. Они могут быть использованы для выполнения ряда служебных функций, как, например, программирование внутреннего УФРПЗУ. Каждый порт является фиксатором-защелкой и может адресоваться как побайтно, так и побитно – конкретные физические адреса при этом указаны выше.

Помимо работы в качестве обычных портов ввода/вывода, линии портов могут выполнять ряд альтернативных функций:

– через порт 0 выводится младший байт адреса, а также выдается и принимается в микроконтроллер байт данных при работе с внешней памятью программ/данных (в мультиплексированном режиме). Задаются данные при программировании внутренней памяти программ и читается ее содержимое;

– порт 1 предназначен для задания младшего байта адреса при программировании и чтении внутренней памяти программ микросхемы;

– через порт 2 выводится старший байт адреса (разряды A8 – A15) внешней памяти программ и данных, а также задаются старшие разряды адреса при программировании и верификации УФРПЗУ;

– для каждого из битов порта 3 имеется ряд альтернативных функций (табл.5):

Таблица 5

Альтернативные функции выводов порта P3

Бит	Мнемоника	Назначение
P3.0	RxD	Вход данных последовательного порта
P3.1	TxD	Выход передатчика последовательного порта
P3.2	INT0	Внешнее прерывание 0
P3.3	INT1	» 1
P3.4	T0	Внешний вход T/C0
P3.5	T1	» T/C1
P3.6	WR	Строб записи во внешнюю память данных
P3.7	RD	Строб чтения из внешней памяти данных

Альтернативные функции портов **P1** и **P2** реализуются только в том случае, если в соответствующий разряд фиксатора-защелки порта записана логическая 1, иначе на соответствующем выводе будет присутствовать 0. То же относится и к портам **P0 – P3**.

Ряд команд микроконтроллера производит **чтение** данных непосредственно с входных линий портов. К ним относятся операции пересылки данных:

MOV A , <port>
MOV Rn , <port>
MOV @Ri , <port>
MOV C , <port.bit>

Имеется набор операций с портами, действующих по принципу «чтение – модификация – запись». Все они производят чтение состояния самой защелки выходного порта, модифицируют это значение и вновь записывают в порт. Пример подобных команд:

ORL <port>, <source>
XRL <port>, <source>
JBC <port.bit>, <addr>

CPL **<port.bit>**
INC **<port>**
DEC **<port>**
DJNZ< **<port> , <addr>**
CLR **<port.bit>**
SETB **<port.bit>**
MOV **<port.bit> , <source>**

Где <source> – операнд-источник – аккумулятор, регистр, косвенно-адресуемая ячейка памяти.

Для избежания ошибок всегда следует учитывать вышеизложенное, поскольку входной уровень порта часто может не соответствовать состоянию защелки. Иногда на выходной защелке присутствует состояние 1 и одновременно на ту же линию порта подается сигнал низкого уровня. При попытке изменить значение выходного бита порта с помощью байтных команд чтения и записи:

MOV A , <port>
ANL A , <mask>
MOV <port> , A

в бит порта будет записана конъюнкция входной линии порта и значения <mask>.

4.2. Последовательный интерфейс микроконтроллеров MCS-51

Через универсальный асинхронный приемопередатчик **(УАПП)** микроконтроллера осуществляется прием и передача информации, представленной последовательным кодом (младшими битами вперед), в полном дуплексном режиме обмена. В состав **УАПП**, называемого часто последовательным портом, входят принимающий и передающий сдвигающие регистры, а также специальный буферный регистр **(SBUF)** приёмопередатчика. Запись байта в буфер приводит к автоматической переписи байта в сдвигающий регистр передатчика и инициирует начало передачи байта. Наличие буферного регистра приемника позволяет совмещать операцию чтения ранее принятого байта с приемом очередного. Если к моменту окончания приема байта предыдущий байт не был считан из **SBUF**, то он будет потерян.

Последовательный интерфейс серии MCS-51 может работать в четырех режимах:

– **режим 0.** Информация передается и принимается через вход **RxD** (линия порта в/в **P3.0**). Через выход передатчика **TxD** (**P3.1**) выдаются синхроимпульсы, стробирующие принимаемые или выдаваемые биты. Формат посылки – 8 бит. Частота приема и передачи – «частота тактирования» – Fclc/12;

– **режим 1.** Информация передается через выход **TxD**, а принимается через **RxD**. Формат посылки – 10 бит (стартовый – 0, 8 информационных и стоповый

– 1). Частота приема и передачи задается программированием **таймера/счетчика 1 (T/C 1)**;

– **режим 2.** Информация передается через выход **TxD**, а принимается через **RxD**. Формат посылки – 11 бит (стартовый - 0, 8 информационных, программируемый девятый бит и стоповый – 1). Девятый бит при передачи транслируется из **SCON TB8**, а при приеме – передается в бит **SCON.RB8**. Частота приема задается программно и может быть равна Fclc/32 или Fclc/64. Девятый бит используется по усмотрению программиста, например, как бит контроля информации по четности или в многопроцессорных системах для идентификации адреса приемника переданного кадра;

– **режим 3** идентичен режиму 2 с тем отличием, что частота приема/передачи программируется таймером/счетчиком 1.

Таблица 6

Регистр SCON управления последовательным интерфейсом

SM0	SM1	SM2	REN	TB8	RB8	TI	RI

Бит	Поз.	Функция присвоения приоритета прерывания
SM0	SCON.7	Бит определения режима работы
SM1	SCON.6	»
SM2	SCON.5	Бит разрешения многопроцессорной работы при SCON.SM2=1: **в режимах 2 и 3** бит RI не активизируется, если девятый бит принимаемых данных равен 0; **в режиме 1** бит RI не активизируется, если не принят стоп-бит равным 1 **В режиме 0** SM2 должен быть равен 0
REN	SCON.4	Бит разрешения приема последовательных данных
TB8	SCON.3	9-й бит передаваемых данных в режимах 2 и 3
RB8	SCON.2	9-й бит принимаемых данных в режимах 2 и 3
TI	SCON.1	Флаг прерывания передатчика устанавливается аппаратно в конце времени выдачи: **в режиме 0** – 8 бита данных; **в режимах 1, 2, 3** – стоп-бита Сбрасывается программно
RI	SCON.0	Флаг прерывания приемника устанавливается аппаратно в конце времени приема: **в режиме 0** – 8 бита данных; **в режимах 1, 2, 3** – стоп-бита

SM0	SM1	Режим	Функция	Скорость передачи*
0	0	0	Сдвиговый регистр	Fclc / 12
0	1	1	8-битовый универсальный асинхронный приемопередатчик	Ftc1
1	0	2	9-битовый УАПП	Fclc / 32,

				Fclc / 64
1	1	3	»	Ftc1

* – Fclc - частота тактирования кристалла;
Ftc1 - частота на выходе таймера / счетчика 1.

Для управления работой последовательного порта применяется **SFR**-регистр **SCON**, формат которого представлен в табл.6. Этот регистр содержит не только управляющие биты, определяющие режим работы последовательного порта, но и девятый бит принимаемых или передаваемых данных (**RB8** и **TB8**) и биты прерывания приемопередатчика (**RI** и **TI**). Прикладная программа путем загрузки в старшие биты регистра **SCON** двухбитного кода определяет режимы работы **УАПП**. Во всех четырех режимах работы передача из **УАПП** инициируется любой командой, в которой буферный регистр **SBUF** указан как получатель байта. Прием в **УАПП** в режиме 0 осуществляется при условии, что **RI**=0 и **REN**=1. В режимах 1,2,3 прием начинается с приходом старт-бита, если **REN**=1.

В бите **TB8** программно устанавливается значение девятого бита данных, который будет передан в режиме 2 или 3. В бите **RB8** фиксируется в режимах 2 и 3 девятый принимаемый бит данных. В режиме 1, если **SM2**=0, в бит **RB8** заносится стоп-бит. В режиме 0 бит **RB8** не используется.

Флаг прерывания передатчика **TI** устанавливается аппаратно в конце периода передачи восьмого бита данных в режиме 0 и в начале периода передачи стоп-бита в режимах 1, 2 и 3. Соответствующая подпрограмма обслуживания прерывания должна сбрасывать бит **TI**.

Флаг прерывания приемника **RI** устанавливается аппаратно в конце периода приема восьмого бита данных в режиме 0 и в середине периода приема стоп-бита в режимах 1,2 и 3. Подпрограмма обслуживания прерывания должна сбрасывать бит **RI**.

Скорость приема/передачи, т.е. частота работы УАПП в разных режимах, определяется различными способами.

В режиме 0 частота передачи зависит только от резонансной частоты кварцевого резонатора $f_0=f_{рез}/12$. За один машинный цикл последовательный порт передает один бит информации.

В режимах 1, 2 и 3 скорость приема/передачи зависит от значения управляющего бита **SMOD** в регистре специальных функций (табл.7).

В режиме 2 частота передачи определяется выражением $f_2=(2^{SMOD}/64)f_{рез}$. Иными словами, при **SMOD**=0 частота передачи равна $(1/64)f_{рез}$, а при **SMOD**=1 равна $(1/32)f_{рез}$.

В режимах 1 и 3 в формировании частоты передачи кроме управляющего бита **SMOD** принимает участие таймер 1. При этом частота передачи зависит от частоты переполнения (OVT1) и определяется следующим образом: $f_{1,3}=(2^{SMOD}/32)f_{OVT1}$.

Таблица 7

Регистр управления энергопотреблением

Символ	Позиция	Наименование и функции
SMOD	PCON.7	Удвоенная скорость передачи. Если бит установлен в 1, то скорость передачи вдвое больше, чем при SMOD=0
-	PCON.6	Не используются
-	PCON.5	-
-	PCON.4	-
GF1	PCON.3	Флаг, специфицируемый пользователем (флаг общего назначения)
GF0	PCON.2	»
PD	PCON.1	Бит пониженной мощности. При установке бита 1 МК переходит в режим пониженного потребления мощности
IDL	PCON.0	Бит холостого хода. Если бит установлен в 1, то МК переходит в режим холостого хода.

Примечание. При одновременной записи 1 в **PD** и **IDL** бит **PD** имеет преимущество. Сброс содержимого РУМ выполняется путем загрузки в него кода 0XXX0000.

Таблица 8

Настройка таймера 1 для управления частотой работы УАПП

Частота приема/передачи		Частота резонатора, МГц	SMOD		Таймер/счетчик 1	
					C/T режим (MODE)	Перезагружаемое число
Режим 0, макс:	1 МГц	12	X	X	X	X
Режим 2, макс:	375 кГц	12	1	X	X	X
Режимы 1, 3:	62,5 кГц	12	1	0	2	0FFH
	19,2 кГц	11,059	1	0	2	0FDH
	9,6 кГц	11,059	0	0	2	0FDH
	4,8 кГц	11,059	0	0	2	0FAH
	2,4 кГц	11,059	0	0	2	0F4H
	1,2 кГц	11,059	0	0	2	0E8H
	137,5 кГц	11,059	0	0	2	1DH
	110 кГц	6	0	0	2	72H
	110 кГц	12	0	0	1	0FEEBH

Прерывание от таймера 1 в этом случае должно быть заблокировано. Сам Т/С1 может работать и как таймер, и как счетчик событий в любом из трех режимов. Однако наиболее удобно использовать режим таймера с автоперезагрузкой (старшая тетрада **TMOD**=0010B). При этом частота передачи определяется выражением $f_{1,3}=(2^{SMOD}/32)f_{рез}/12)(256-(\mathbf{TH1}))$. В табл. 8 приводится описание способов настройки Т/С1 для получения типовых частот передачи данных через УАПП.

Контрольные вопросы

1. Сформулируйте назначение параллельных портов ввода/вывода ОЭВМ.
2. Перечислите альтернативные функции линий портов ввода/вывода ОЭВМ.
3. Сформулируйте особенности программирования ввода/вывода с использованием параллельных портов.
4. Сформулируйте назначение последовательного интерфейса ввода/вывода ОЭВМ.
5. Дайте характеристику режимов работы последовательного интерфейса.
6. Приведите примеры программирования режимов работы последовательного интерфейса.

5. Организация и управление таймерами/счётчиками микроконтроллера MCS-51

Таймеры/счетчики (Т/С) предназначены для подсчета внешних событий, организации программно управляемых временных задержек и измерения временных интервалов. В микроконтроллере реализованы два шестнадцатиразрядных многорежимных таймера/счётчика.

Таблица 9

Регистр режимов таймеров/ счетчиков TMOD

GATE1	C/T1	M1.1	M0.1	GATE0	C/T'0	M1.0	M0.0

Бит	Позиция	Функция
GATE1	TMOD.7	Разрешение управления таймером Т/С1 от внешнего вывода INT1. При GATE1=1 — управление разрешено
C/T'1	TMOD.6	Определение работы Т/С1 как: **C/T1 = 0 —** таймера; **C/T1 = 1 —** счетчика
M1.1 M0.1	TMOD.5 TMOD.4	Определение режима работы Т/С1
GATE0	TMOD.3	Разрешение управления таймером Т/С0 от внешнего вывода INT0 при GATE0=1 – управление разрешено

C/T'0	TMOD.2	Определение работы T/C0 как: **C/T0 = 0** — таймера; **C/T0 = 1** — счетчика
M1.0 M0.0	TMOD.1 TMOD.0	Определение режима работы T/C0

M1.X	M0.X	Режим
0	0	0
0	1	1
1	0	2
1	1	3

В состав T/C входят следующие узлы:

– ***восьмиразрядный регистр режимов TMOD***. Предназначен для установки режима работы T/C и определения источника его тактирования. Значения разрядов данного регистра приведены в табл.9.

Таблица.10

Регистр управления таймера/счетчика TCON

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

Бит	Позиция	Функции
TF1	TCON.7	**Бит переполнения T/C1** Устанавливается по переходу счетчика из FFh в 00h; при разрешении прерывания от T/C. Установка флага вызывает прерывание; сбрасывается аппаратно при передаче управления обработчику прерывания. Флаг может быть установлен программно, также доступен по чтению
TR1	TCON.6	**Бит включения T/C1** Устанавливается и сбрасывается программно
TF0	TCON.5	**Бит переполнения T/C0** (см. бит TF1)
TR0	TCON.4	**Бит включения T/C0** (см. бит TR1)
IE1	TCON.3	**Флаг запроса внешнего прерывания по входу INT1** Устанавливается аппаратно по прерыванию или программно, инициируя вызов обработчика соответствующего прерывания. Сброс выполняется аппаратно только в том случае, когда прерывание было вызвано фронтом сигнала (см. бит вида прерывания); в случае, если прерывание вызвано уровнем сигнала на входе INT1, сброс флага должен осуществляться посредством программного воздействия на источник прерывания для снятия им запроса. Доступен также для чтения
IT1	TCON.2	**Бит вида прерывания по входу INT1** 1 — прерывание по уровню (низкому) 0 — прерывание по фронту (из 1 в 0)

		Доступен для чтения и записи
IE0	TCON.1	**Флаг запроса внешнего прерывания по входу INT0** (см. бит IE1)
IT0	TCON.0	**Бит вида прерывания по входу INT0** (см. бит IT1)

– ***два 16-разрядных регистра T/C0 и T/C1*.** Выполняют функцию хранения содержимого счета. Состоят из четырех 8-разрядных регистров TH0, TH1, TL0, TL1. Регистры TH0 и TH1 содержат старший байт значения T/C, а TL0 и TL1 – младший. Доступны как для записи – занесение начальной величины счета, так и для чтения – контроля их содержимого. В процессе счета содержимое регистров инкрементируется. Переполнение счетчиков (т.е. их переход из FFh в 00h) может служить признаком окончания счета и инициализировать прерывание от таймеров;

– ***восьмиразрядный регистр управления TCON.*** Предназначен для приема и хранения управляющего слова T/C. Значения разрядов данного регистра приведены в табл.10;

– **схема инкремента**. Предназначена для увеличения в каждом машинном цикле (с частотой тактирования Fosc/12) содержимого тех T/C, для которых установлен режим таймера и счет разрешен, либо тех T/C, для которых установлен режим счетчика и на входе T0 или T1 имеется счетный импульс (переход из 1 в 0). Счетные входы аппаратно проверяются в каждом машинном цикле. Инкрементированное значение заносится в T/C в каждом машинном цикле, следующим за тем, в котором был обнаружен счетный импульс. Таким образом, минимальный период счета будет равен двум машинным циклам, т.е. максимальная частота счета равна Fosc/24;

– **схема фиксации сигналов INT0, INT1, T0, T1.**Представляет собой 4 триггера, запоминающее состояние сигналов INT0, INT1, T0 и T1 в каждом машинном цикле;

– **схема управления флагами**. Генерирует флаги переполнения T/C и запросов внешних прерываний.

– **логика управления T/C**. Синхронизирует работу регистров T/C с работой ЦП микроконтроллера.

Имеются 4 режима работы T/C микроконтроллера, определяемые установкой соответствующих битов регистра TMOD (табл.9). Режимы 0,1 и 2 полностью идентичны для обоих T/C. Установка в режим 3 T/C0 влияет на режим работы T/C1.

Режим 0. T/C являются устройствами на базе 13-разрядных регистров, образованных соответствующими регистрами Thx и пятью младшими разрядами регистров TLx (3 старших разряда TL0 и TL1 являются незначащими). Регистры TLx выполняют функцию делителя на 32. Счет начинается при установке бита TRx регистра TCON в состояние 1. Установка в 1 бита GATEx регистра TMOD разрешает управление T/C извне. Установка в 0 бита TRx регистра TCON запрещает счет независимо от состояния других битов. Бит C/Tx определяет работу T/C как таймеров (0) или как счетчиков (1).

Режим 1. Данный режим отличается от предыдущего только тем, что в счетчике используется 16-разрядный, а не 13-разрядный регистр. Работа Т/С в этом случае идентична.

Режим 2. В этом режиме используются 8-разрядные регистры на основе TL0 (в Т/С0) и TL1 (в Т/С1). При переполнении этих регистров в процессе счета происходит перезагрузка их значением из регистров TH0 или TH1. Эти регистры загружаются программно, и процесс перегрузки из TH0 (или TH1) в TL0 (или TL1) не влияет на их содержимое. Назначение битов управления в данном режиме такое же как и в режимах 0 и 1.

Режим 3. В этом режиме работает только Т/С0. Счетчик Т/С1 в этом режиме заблокирован и сохраняет содержимое своих регистров TL1 и TH1 (как при TR1=0). Счетчик Т/С0 представлен двумя независимыми 8-разрядными регистрами TL0 и TH0 **.** За устройством на базе регистра TL0 закреплены все биты управления, флаги и входные линии Т/С0. Устройство на базе TH0 может работать только в режиме таймера и использует некоторые управляющие биты и флаги Т/С (например, при переполнении TH0 происходит установка TF1, а для включения используется бит TR1). Остальные биты счетчика Т/С1 с работой TH0 не связаны. Установка Т/С0 в режим работы 3 приводит к лишению Т/С1 управляющего бита включения TR1. По этой причине при нахождении Т/С0 в 3-м режиме, обнуляется, а во 2-м режиме — перезагружается, не устанавливая флаг.

Контрольные вопросы

1. Сформулируйте назначение таймеров/счетчиков ОЭВМ.
2. Дайте характеристику регистров TMOD и TCON ОЭВМ.
3. Дайте характеристику режимов работы таймеров/счетчиков ОЭВМ.
4. Приведите примеры программирования режимов работы таймеров/счетчиков.

6. Организация прерываний в микроконтроллерах серии MCS-51

6.1. Структура прерываний

Механизм прерываний в МК позволяет автоматически реагировать на внешние и на внутренние события (переполнение таймеров/счетчиков, завершение последовательного обмена). Механизм обработки прерывания при обнаружении запроса прерывания представлен на рис.9.

Архитектура МК MCS-51 обеспечивает поддержку пяти источников прерываний: двух внешних прерываний, двух прерываний от таймера, прерывания от последовательного порта.

Каждое из внешних прерываний **INT0**, **INT1** может быть активизировано по уровню («0») или по фронту (переход из 1 в 0) сигналов на выводах МК **P3.2**, **P3.3**, что определяется состоянием битов **IT0** и **IT1** регистра **TCON**. При поступлении запроса внешнего прерывания **INTx** устанавливается флаг **IEx**

регистра **TCON**. Установка флагов **IEx** в регистре **TCON** вызывает соответствующее прерывание. Очистка флага **IEx** производится следующим образом: при прерывании по фронту **IEx** сбрасывается аппаратно (автоматически внутренними средствами МК) при обращении к соответствующей подпрограмме обработки прерывания; при прерывании по уровню флаг очищается при снятии запроса внешнего прерывания, т.е. в **IEx** отслеживается состояние вывода **INTx**.

Чтобы внешнее прерывание по уровню было распознано, необходимо, чтобы низкий уровень на выводе **INTx** удерживался в течение не менее 12 периодов сигнала тактовой частоты МК. Это объясняется тем, что проверка выводов **INT0**, **INT1** выполняется внутренними аппаратными средствами МК один раз в каждом машинном цикле. В случае внешнего прерывания по фронту флаг **IEx** будет установлен, если две последовательные проверки входа **INTx** покажут в одном машинном цикле 1, а в следующем – 0. Поэтому, если внешнее прерывание активизируется по переходу из состояния высокого уровня в состояние низкого уровня, то минимум одному машинному циклу низкого уровня должен предшествовать минимум один машинный цикл высокого уровня на выводе **INTx**. Если внешнее прерывание активизируется по уровню, запрос должен удерживаться до начала обслуживающей подпрограммы и сниматься до завершения этой подпрограммы для предотвращения повторного обслуживания.

Прерывания от таймеров/счетчиков вызываются установкой флагов **TF0** и **TF1** регистра **TCON**, которые устанавливаются при переполнении соответствующих регистров таймеров/счетчиков (за исключением режима 3). Очистка флагов **TF0** и **TF1** производится внутренней аппаратурой МК при переходе к подпрограмме обслуживания прерывания.

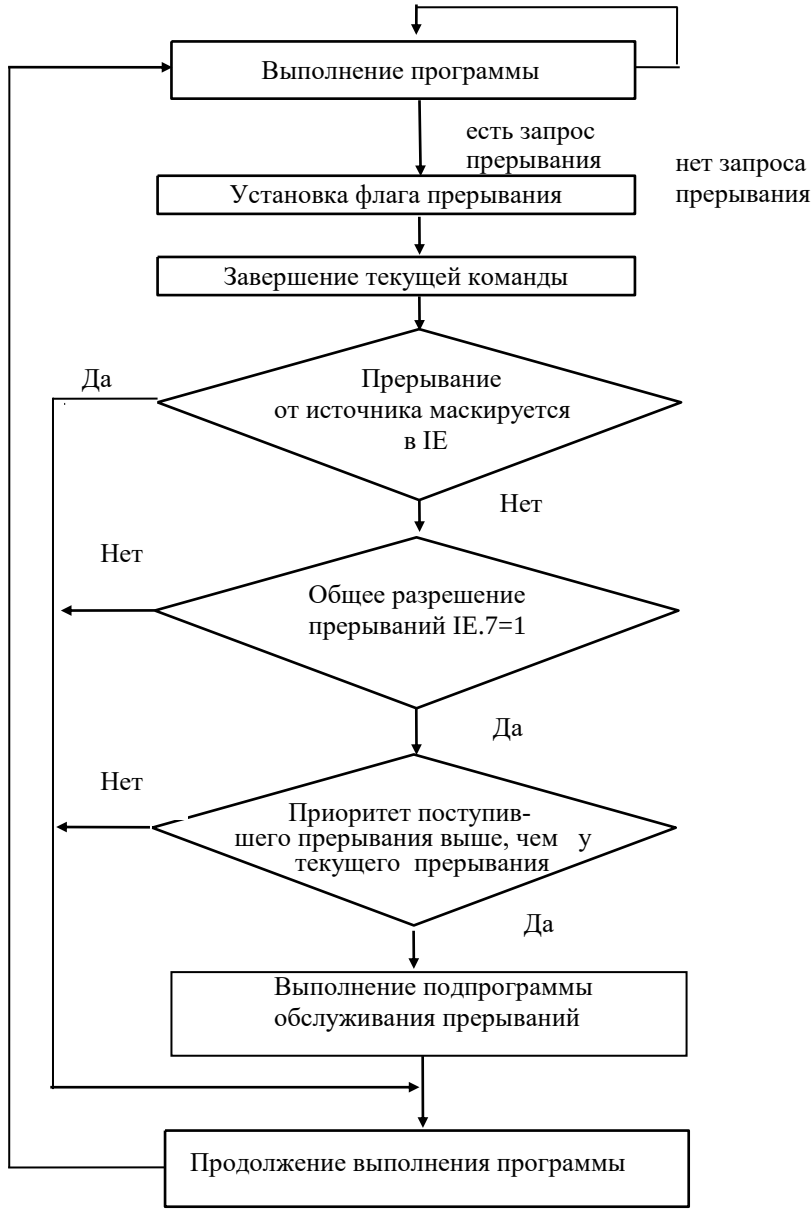


Рис.9. Обобщенная схема обработки прерывания

Прерывание от последовательного порта вызывается установкой флага прерывания приемника **RI** или флага прерывания передатчика **TI** в регистре **SCON**. В отличие от всех остальных флагов, **RI** и **TI** сбрасываются только

программным путем обычно в пределах подпрограммы обработки прерывания, где определяется, какому из флагов **RI** или **TI** соответствует прерывание.

Каждый из перечисленных источников прерываний может быть индивидуально разрешен или запрещен установкой или сбросом соответствующего бита в регистре разрешения прерываний **IE** (Interrupt Enable). Регистр **IE** содержит также бит **EA**, сброс которого в 0 запрещает сразу все прерывания. Необходимым условием прерывания является его разрешение в регистре **IE**. Формат и описание регистра разрешения прерываний приведены в табл.11.

Таблица 11
Регистр IE

EA	**—**	**—**	**ES**	**ET1**	**EX1**	**ET0**	**EX0**

1 - прерывания разрешены
0 - прерывания запрещены

Бит	**Поз.**	**Функция запрета / разрешения прерывания**
EA	IE.7	Запрещает прерывания от всех источников
—	IE.6	Зарезервировано для дальнейшего использования
—	IE.5	»
ES	IE.4	Последовательный порт
ET1	IE.3	Переполнение 1-го таймера
EX1	IE.2	Внешнее прерывание INT1
ET0	IE.1	Переполнение 0-го таймера
X0	IE.0	Внешнее прерывание INT0

Все биты, которые вызывают прерывания (**IE0**, **IE1**, **TF0**, **TF1**, **RI**, **TI**), могут быть программно установлены или сброшены с тем же результатом, что и в случае их аппаратной установки или сброса. Т.е. прерывания могут программно вызываться или ожидающие обслуживания прерывания могут программно ликвидироваться. Кроме того, прерывания по **INT0**, **INT1** могут вызываться программой установки **P3**.2=0 и **P3**.3=0, как показано в приведенном ниже примере:

MAIN: MOV IE, #00000101B; разрешение прерывания от **INT0**, **INT1**.
MOV IP, #04H; присвоение **INT1** старшего приоритета.
SET EA; общее разрешение прерывания.
MOV P3, #11110011B; имитация внешних прерываний.

В предложенном примере запросы прерывания **INT0** и **INT1**, имеющие различный приоритет, поступают одновременно. При этом обслуживается прерывание с высшим приоритетом.

В случае, когда прерывание по **INTx** (x=0,1) вызывается уровнем сигнала на соответствующем входе МК, флаг **IEx** (x=0,1) при переходе к подпрограмме обработки прерывания автоматически сбрасывается, а затем, если соответствующий вывод МК **P3.2** или **P3.3** все еще находится в состоянии логического 0, вновь устанавливается. Поэтому, в случае, когда прерывание по

входам **INT0**, **INT1** вызывается уровнем, программная установка в 1 флагов **IE0**, **IE1** вызовёт прерывание, после чего соответствующий флаг **IEx** (x=0,1) будет автоматически сброшен при переходе к подпрограмме обработки прерывания.

Флаги **IE0**, **IE1**, **TF0**, **TF1**, **RI**, **TI** устанавливаются независимо от того разрешено или нет соответствующее прерывание в регистре **IE**.

6.2. Приоритеты прерываний

Каждому источнику прерывания может быть индивидуально присвоен один из двух уровней приоритета: высокий или низкий. Выполняется это установкой (высокий уровень приоритета) или сбросом (низкий уровень приоритета соответствующего бита в регистре приоритетов прерываний IP (табл.12). Программа обработки прерывания с низким уровнем приоритета может быть прервана запросом прерывания с высоким уровнем приоритета, но не может быть прервана другим запросом прерывания с низким уровнем приоритета. Программа обработки прерывания с высоким уровнем приоритета не может быть прервана никаким другим запросом прерывания ни от одного из источников. Если два запроса с разными уровнями приоритета приняты одновременно, сначала будет обслужен запрос с высоким уровнем приоритета. Если одновременно приняты запросы с одинаковым уровнем приоритета, обработка их будет производиться в порядке, задаваемом последовательностью внутреннего опроса флагов прерываний. Таким образом, в пределах одного приоритетного уровня существует еще одна структура приоритетов:

Источник		Приоритет внутри уровня
1.	IE0	(высший)
2.	TF0	
3.	IE1	
4.	TF1	
5.	RI+TI	(низший)

Необходимо особо подчеркнуть, что структура «Приоритет внутри уровня» работает только в тех случаях, когда определяется последовательность обслуживания запросов на прерывания, которые приняты одновременно и при этом имеют одинаковый уровень приоритета.

Таблица.12

Регистр IP

—	—	—	**PS**	**PT1**	**PX1**	**PT0**	**PX0**

1 - высший приоритет

0 - низший приоритет

Бит	Поз.	Функция присвоения приоритета прерывания
—	IP.7	Зарезервировано для дальнейшего использования
—	IP.6	»
—	IP.5	»

ES	IP.4	Последовательный порт
ET1	IP.3	Переполнение 1-го таймера
EX1	IP.2	Внешнее прерывание INT1
ET0	IP.1	Переполнение 0-го таймера
EX0	IP.0	Внешнее прерывание INT0

6.3. Обработка прерываний и время отклика

Уровни на выводах INT0 и INT1 инвертируются и защелкиваются в флаги прерывания IE0 и IE1 в каждом машинном цикле. Устанавливаются флаги прерываний последовательного порта RI и TI. Флаги TF0 и TF1 таймеров/ счетчиков устанавливаются в машинном цикле, в котором происходит переполнение T/C. Анализ (опрос) флагов выполняется внутренними средствами МК в следующем после установки (защелкивания) флагов машинном цикле (цикл опроса флага). И только после выполнения последнего цикла текущей команды производится аппаратный вызов соответствующей подпрограммы обслуживания, эквивалентной команде LCALL.

Обращение к подпрограмме обслуживания задерживается (блокируется аппаратный LCALL) при выполнении хотя бы одного из следующих условий:

– уже производится обработка прерывания с таким же или высшим приоритетом;

– текущий машинный цикл (цикл опроса флага) не является последним циклом выполняемой команды;

– выполняемая команда текущей программы является командой RETI или любой командой обращения к регистрам IE, IP .

В последнем условии после окончания одной из вышеуказанных команд обязательно выполнится еще минимум одна команда текущей программы перед вызовом подпрограммы обслуживания прерывания.

Флаг прерывания, установленный во время действия блокировки прерывания по одному из трех указанных выше условий и сброшенный до их снятия, не вызовет обслуживания соответствующего запроса прерывания.

Если запрос прерывания с более высоким уровнем приоритета зафиксируется во время аппаратного вызова подпрограммы обслуживания, то по окончании процедуры текущего вызова сразу же начнет выполняться процедура аппаратного вызова по поступившему запросу.

Аппаратно-реализуемая команда **LCALL** загружает содержимое счетчика команд **PC** в стек (при этом ССП в стек не записывается), после чего записывает в **PC** адрес соответствующей подпрограммы обработки прерывания:

Источник прерывания	Адрес подпрограммы (вектор прерывания)
IE0	0003h
TF0	000Bh
IE1	0013h
TF1	001Bh

TI+RI	0023h

При выполнении аппаратно-реализуемой команды **LCALL** в ячейку стека с младшим адресом загружаются разряды 0-7 счетчика команд, а в следующую ячейку стека – разряды 8-15 счетчика команд.

Подпрограмма обслуживания прерывания продолжается до выполнения команды **RETI**. Команда **RETI** восстанавливает состояние логики прерывания и загружает в счетчик команд **PC** 2 байта адреса возврата из двух верхних ячеек стека. Восстановление состояния логики прерывания заключается в следующем: при переходе по вектору на подпрограмму обработки прерывания автоматически до выполнения команды **RETI** независимо от состояния бит регистра IE запрещаются все прерывания с уровнем приоритета, равным уровню приоритета обслуживаемого прерывания, т.е. вложенные прерывания с равными уровнями приоритета невозможны. Команда **RETI** снимает этот запрет. При использовании команды **RET** восстанавливается только состояние счетчика команд, т.е. происходит возврат в прерванную программу. Состояние логики прерывания команда **RET** не меняет, т.е. логика управления обслуживанием прерываний по-прежнему считает, что продолжает обслуживаться прерывание, подпрограмма обработки которого была закончена командой **RET**.

Контрольные вопросы

1. Дайте общую характеристику системы прерываний ОЭВМ.
2. Сформулируйте назначение и покажите особенности програм-мирования регистра разрешения прерываний IE.
3. Сформулируйте назначение и покажите особенности програм-мирования регистра приоритетов прерываний IP.
4. Охарактеризуйте механизм вызова подпрограмм обслуживания прерываний.

7. Основы программирования на языке Ассемблера микроконтроллера MCS-51

7.1. Правила записи программ на языке Ассемблера

Исходный текст программы на языке Ассемблера имеет определенный формат. Каждая команда (и псевдокоманда) представляет собой строку четырехзвенной конструкции:

МЕТКА ОПЕРАЦИЯ ОПЕРАНД(Ы) КОММЕНТАРИЙ

Звенья (поля**)** могут отделяться друг от друга произвольным числом пробелов.

Метка. В поле метки размещается символическое имя ячейки памяти, в которой хранится отмеченная команда или операнд. Метка представляет собой буквенно-цифровую комбинацию, начинающуюся с буквы. Используются

только буквы латинского алфавита. Ассемблер МК допускает использование в метках символа подчеркивания (_). Длина метки не должна превышать шесть символов. Метка всегда завершается двоеточием (:).

Псевдокоманды Ассемблера не преобразуются в двоичные коды, а потому не могут иметь меток. Исключение составляют псевдокоманды резервирования памяти и определения данных (DS, DB, DW). У псевдокоманд, осуществляющих определение символических имен, в поле метки записывается определяемое символическое имя, после которого двоеточие не ставится.

В качестве символических имен и меток не могут быть использованы мнемокоды команд, псевдокоманд и операторов ассемблера, а также мнемонические обозначения регистров и других внутренних блоков микроконтроллера.

Операции. В поле операции записывается мнемоническое обозначение команды или псевдокоманды Ассемблера, которое является сокращением (аббревиатурой) полного английского названия выполняемого действия. Например: MOV – move – переместить, JMP – jump – перейти, DB – define byte – определить байт.

Для МК используется строго определенный и ограниченный набор мнемонических кодов. Любой другой набор символов, размещенный в поле операции, воспринимается Ассемблером как ошибочный.

Операнды. В этом поле определяются операнды (или операнд), участвующие в операции. Команды ассемблера могут быть без-, одно- или двухоперандными. Операнды разделяются запятой (,) .

Операнд может быть задан непосредственно или в виде его адреса (прямого или косвенного). Непосредственный операнд представляется числом (**MOV A, #15**) или символическим именем (**ADDC A, #OPER2**) с обязательным указанием префикса непосредственного операнда (#). Прямой адрес операнда может быть задан мнемоническим обозначением (**MOV A, P1**) , числом (**INC 40**), символическим именем (**MOV A, MEMORY**). Указанием на косвенную адресацию служит префикс @. В командах передачи управления операндом может являться число (**LCALL 0135H**), метка (**JMP LABEL**), косвенный адрес (**JMPP @A**) или выражение (**JMP ¤ – 2**, где ¤ – текущее содержимое счетчика команд).

Используемые в качестве операндов символические имена и метки должны быть определены, а числа представлены с указанием системы счисления, для чего используется специальное обозначение (буква, стоящая после числа): B – для двоичной, Q – для восьмиричной, D – для десятичной и H – для шестнадцатиричной. Число без такого обозначения по умолчанию считается десятичным.

Ассемблер МК допускает использование выражений в поле операндов, значение которых вычисляются в процессе трансляции.

Выражение представляет собой совокупность символических имен и чисел, связанных операторами ассемблера. Операторы ассемблера обеспечивают выполнение арифметических («+» – сложение, «-» – вычитание, * – умножение,

/ – целое деление, MOD – деление по модулю) и логических (OR – ИЛИ, AND – И, XOR – исключающее ИЛИ, NOT – отрицание) отрицаний в формате двухбайтных слов.

Например, запись **ADD A, #((NOT 13)+1)** эквивалентно записи **ADD A, #0F3H** и обеспечивает сложение содержимого аккумулятора с числом - 13, представленным в дополнительном коде.

Комментарий. Поле комментария может быть использовано программистом для текстового и символьного пояснения логической организации прикладной программы. Поле комментария полностью игнорируется ассемблером, а потому в нем допустимо использовать любые символы (в некоторых версиях ассемблера только букв английского алфавита). По правилам языка Ассемблера поле комментария начинается после точки с запятой (;).

В результате трансляции должна быть получена карта памяти программ, где каждой ячейке памяти поставлен в соответствие хранящийся в ней код.

7.2. Директивы языка

Ассемблирующая программа транслирует исходную программу в объектные коды. Транслирующая программа берет на себя многие из рутинных задач, таких, как присвоение действительных адресов, преобразование чисел, присвоение действительных значений символьным переменным и т.п., однако программист должен указать ей некоторые параметры: начальный адрес прикладной программы, конец ассемблируемой программы, форматы данных и т. д. Вся эта информация вносится в исходный текст прикладной программы в виде псевдокоманд (директив) ассемблера, которые только управляют процессом трансляции и не преобразуются в коды объектной программы.

В результате трансляции должна быть получена карта памяти программ, где каждой ячейке памяти поставлен в соответствие хранящийся в ней код.

В языке Ассемблер имеются следующие директивы: начальной установки счетчика адреса Ассемблера ORG; определения имен-синонимов; определения данных DB и DW; резервирования памяти DS; окончания исходной программы END.

Поле метки в формате предложения при записи директив EQU и SET запрещено, при записи всех остальных директив – может использоваться для определения метки в программе.

Основной внутренней переменной Ассемблера является счётчик ячеек (адресов), который при ассемблировании выполняет функцию, аналогичную функции программного счётчика при выполнении программы. Счётчик адресов сообщает Ассемблеру адрес следующей ячейки памяти, которая предназначена для размещения следующего байта команды или данных.

Текущее значение счётчика адресов может быть принудительно изменено с помощью директивы ORG.

Директива ORG

Формат: ORG ⟨EXP⟩

выражение языка.

Все термы выражения ⟨EXP⟩ должны быть самоопределёнными или предварительно определенными. По директиве ORG в счетчик адреса Ассемблера записывается значение выражения ⟨EXP⟩. Данная директива может быть использована для первоначальной установки счетчика адреса, например:

ORG 100h

или для изменения его текущего значения, например:

ORG ¤ + 100h

На уровне языка Ассемблер содержимое счетчика адреса доступно в выражении посредством терма «Обозначение счетчика адреса» (¤).

Пример использования директивы ORG:

START: ORG 3FFH

Директива EQU

Формат: ⟨NAME⟩ EQU ⟨EXP⟩ | ⟨RC⟩

имя программно-доступного элемента МК

выражение языка

имя-синоним

По данной директиве выражению ⟨EXP⟩ или имени программно-доступного элемента МК ⟨RG⟩ присваивается имя-синоним ⟨NAME⟩. Выражение ⟨EXP⟩ может содержать термы любого вида определяемости. Диапазон изменения значения выражения может быть в пределах от 0 до 65 535. В качестве термов в выражении допускается использовать имена-синонимы. Ссылки на послеопределенные имена-синонимы программно-доступных устройств МК запрещены. Имя-синоним, объявленное директивой EQU, не может быть переопределено в программе каким-либо способом.

Пример использования директивы EQU:

VAL EQU 55h

Директива SET

Формат: ⟨NAME⟩ SET ⟨EXP⟩ | ⟨RC⟩

имя программно-доступного элемента МК

выражение языка

имя-синоним

Данная директива аналогична директиве EQU за исключением того, что имя-синоним ⟨NAME⟩ может быть переопределено другой директивой SET.

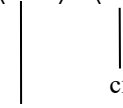
Допускается итеративное определение имени-синонима.

Пример использования директивы SET:

```
IMMED   SET    10101B
MOV  A, #IM
IMMED   SET    IM + 6
MOV  A, #IM
```

Директива DB

Формат: DB ⟨EXP⟩ | ⟨STP⟩ [, ...]



Выражение ⟨EXP⟩ в директиве DB может иметь термы любого вида определяемости. Диапазон изменения значения выражения должен быть от 0 до 255. Директива DB используется для определения 8-битных данных (байтов данных) в памяти программ, которые рассматриваются как константы.

Директива DB выполняется следующим образом: определяется значение выражения ⟨EXP⟩ или значение кода очередного символа символьной строки ⟨STR⟩. Это значение рассматривается как байт данных, который последовательно включается ассемблером в объектную программу.

Максимальное количество аргументов в директиве равно восьми (при условии, что все аргументы могут быть размещены в одной строке предложения).

Примеры использования директивы DB:

```
MSG:    DB 'ABC', '12', 0Dh  ;  41h,42h,43h,31h,32h,0Dh
NULL:   DB                   ;  Данные не генерируются
        DB 5+2, 5-2          ;  07h, 03h
```

Директива DW

Формат: DW ⟨EXP⟩ [, ...]

Выражение ⟨EXP⟩ в директиве может иметь термы любого вида определяемости. Диапазон изменения значения выражения может быть от 0 до 65 535.

Директива DW используется для определения 16-битных данных (слова данных) в памяти программ, которые рассматриваются как константы.

Директива выполняется следующим образом: определяется значение выражения ⟨EXP⟩. Это значение рассматривается как слово данных, которое побайтно включается ассемблером в объектную программу, причем старший байт включается в объектную программу первым, а младший – вторым.

Максимальное количество аргументов в директиве равно восьми (при условии, что все аргументы могут быть размещены в одной строке предложения).

Пример использования директивы DW:

```
ADDR:  DW  0125h                ;  01h, 25h
PAGES: DW  0, 100h, 200h        ;  00h, 00h, 01h, 00h, 02h,00h
STP:   DW  'AB', '1'            ;  41h, 42h, 00h, 31h
```

Директива DS

Формат: DS ⟨EXP⟩

выражение языка

В выражение ⟨EXP⟩ могут быть включены только самоопределенные и предварительно определенные термы. Диапазон изменения значения выражения должен быть в пределах от 0 до 4095. Директива DS используется для резервирования области памяти программ под коды команд или данные (константы). Содержимое этой области Ассемблер не определяет.

Директива выполняется следующим образом.

Вычисляется значение выражения ⟨EXP⟩.Полученное значение прибавляется по модулю 2^{12} к содержимому счетчика адреса Ассемблера.

Метка директивы идентифицирует первый байт зарезервированной области памяти. Если значение выражения ⟨EXP⟩ равно нулю, то область памяти не резервируется. Метка директивы идентифицирует код команды или данных (константы), непосредственно генерируемый после этой директивы.

Пример использования директивы DS:

```
Директива                 Содержимое счетчика адреса
   CONST1:  DS 10H              ; 0100h
   CONST2:  DS  5               ; 0110h
            NOP                 ; 0115h
```

Директивы DATA, XDATA, BIT

Ассемблер позволяет определить символическое имя как адрес внутренних (директива DATA), внешних (директива XDATA) данных или адрес бита (директива BIT).

Формат: DATA <EXP>

XDATA <EXP>
BIT <EXP>
|
выражение языка

Примеры использования директив:

FLAGS DATA 25h ; определяется символическое имя
ERROR_FLAG BIT FLAGS.3 ; ERROR_FLAG как третий бит ячейки ОЗУ с адресом 25h

Директива END

Формат: END [⟨EXP⟩]
|
выражение языка

В выражение ⟨EXP⟩ могут быть включены термы любого вида определяемости.

Данная директива определяет конец исходной программы. Если выра-жение ⟨EXP⟩ указано в директиве, то его значение определяет адрес точки входа в программу. Если выражение ⟨EXP⟩ отсутствует в директиве, то адрес точки входа в программу предполагается равным 000h.

В исходной программе должна использоваться только одна директива END, причем она должна быть последним оператором исходной программы. Предложения, следующие за директивой END, Ассемблером игнорируются.

Пример использования директивы END:

```
START: CLR A
       ...
       JMP START
       END START
```

7.3. Система команд языка Ассемблер микроконтроллера

Система команд MCS-51 поддерживает единый набор инструкций, который предназначен для реализации алгоритмов управления исполни-тельными устройствами. Существует возможность использования быстрых методов адресации к внутреннему ОЗУ, осуществление битовых операций над небольшими структурами данных. Имеется развернутая система адресации однобитовых переменных, как самостоятельного типа данных, позволяющая использовать отдельные биты в логических и уп-равляющих командах булевой алгебры. Ниже представлен обзор инструкций с кратким описанием их использования. Все ссылки на язык ассемблера приводятся применительно к продукту **«Intel's MCS-51 Macro Assembler ASM51».**

7.3.1. Режимы адресации

Набор команд MCS-51 поддерживает следующие режимы адресации:

Прямая адресация (Direct addressing). Операнд определяется 8-битным адресом в инструкции. Прямая адресация используется только для внутренней памяти данных и регистров SFR.

Косвенная адресация (Indirect addressing). В этом случае инструк-ция адресует регистр, содержащий адрес операнда. Данный вид адреса-ции используется для внешнего и верхней половины внутреннего ОЗУ.

Для указания 8-битных адресов могут использоваться регистры R0 и R1 выбранного регистрового банка или указатель стека **SP.** Для 16-битной адресации используется только «**регистр указателя данных**» (**DPTR**).

Регистровые инструкции (Register instruction). Регистры R0-R7 текущего регистрового банка, могут быть адресованы через конкретные инструкции, содержащие 3-битовое поле, указывающее номер регистра в самой инструкции. Эти команды эффективны с точки зрения размера кода, поскольку не содержат адреса байта. Во время их выполнения осущест-вляется доступ к одному из восьми регистров выбранного банка. Посредством битов выбора банка, в PSW может осуществляться выбор одного из четырех имеющихся регистровых банков (в момент выполнения программы).

Некоторые инструкции используют индивидуальные регистры. Например, операции с аккумулятором, DPTR и т.д. Данные регистры не имеют адреса, указывающего не них; это заложено в код операции.

Непосредственная адресация (Immediate addressing). Константа может находиться в программе за кодом операции, например:

```
MOV  A , #100
```

Команда производит загрузку аккумулятора десятичным числом 100. Применительно к языку ассемблера, числа также могут быть представлены в двоичном, десятичном и шестнадцатиричном выражении, например 64h.

Индексная адресация (Indexed addressing). Индексная адресация может использоваться только для доступа к программной памяти и толь-ко в режиме чтения. В этом режиме осуществляется просмотр таблиц в памяти программ. Шестнадцатибитовый регистр (DPTR или программный счетчик) указывает базовый адрес требуемой таблицы, а аккумулятор – на точку входа в нее. Адрес таблицы располагается в программной памяти и складывается из значений аккумулятора и регистра.

Другой метод индексной адресации предназначен для «**перехода по выбору**» (Case jump). При этом адрес инструкции перехода вычисляется как сумма базового указателя и аккумулятора.

7.3.2. Арифметические и логические инструкции

Список арифметических инструкций представлен в табл. 13, где показаны режимы адресации, использующие каждую инструкцию доступа к байтовому операнду. Например, команда сложения **ADD** может быть записана следующим образом:

```
ADD  A , 7Fh
ADD  A , @R0
```

ADD A , R7
ADD A , # 127

Время выполнения команд указано в таблице для частоты тактирования – 12МГц. Все арифметические инструкции выполняются за 1 мкс за исключением команды **INC DPTR**, - требующей 2 мкс, а также умножения и деления, выполняемых за 4 мкс. Отметим, что любой байт во внутренней памяти данных может быть инкрементирован и декрементирован без использования аккумулятора. Одна из инструкций инкремента – **INC** – оперирует с указателем данных – регистром **DPTR**, использующимся для получения 16-разрядного адреса внешней памяти данных.

Инструкция **MUL AB** производит умножение данных в аккумуляторе на данные, находящиеся в регистре B, помещая 16-битное произведение в регистры A и B.

Инструкция **DIV AB** делит содержимое аккумулятора на значение в регистре B, оставляя остаток в B, а частное – в аккумуляторе. Эта команда чаще используется для преобразования оснований чисел и сдвигов, чем для «арифметических» операций. В операциях сдвига деление числа на 2^n сдвигает его на n бит вправо.

Инструкция **DA A** предназначена для двоично-десятичных (BCD) арифметических операций. В BCD – операциях команды ADD и ADDC всегда следуют за командой DA A, что обеспечивает сохранение результата в представлении BCD. Однако данная команда не дает преобразование двоичного числа в двоично-десятичное, а обеспечивает правильный результат только в случае, когда применяется в качестве второго шага при сложении двух BCD байт.

Таблица 13

Арифметические инструкции MCS-51

Мнемоника	Действие	Режим адресации	Время вып., мкс
ADD A, byte	A=A+byte	Dir, Ind, Reg, Imm	1
ADDC A, byte	A=A+byte+C	Dir, Ind, Reg, Imm	1
SUBB A, byte	A=A - byte –C	Dir, Ind, Reg, Imm	1
INC A	A=A+1	Аккумулятор	1
INC byte	byte=byte+1	Dir, Ind, Reg	1
INC DPTR	DPTR=DPTR+1	Регистр DPTR	2
DEC A	A=A – 1	Аккумулятор	1
DEC byte	byte=byte – 1	Dir, Ind, Reg	1
MUL A B	B:A=B*A	Аккумулятор и B	4
DIV A B	A=lnt(A/B); B=Mod(A/B)	»	4
DA A	Десятичная коррекция	Аккумулятор	1

* Методы адресации: Dir – прямая, Ind – косвенная, Reg – регистровая, Imm – непосредственная.

В табл. 14 показаны все логические операции ассемблера MCS-51 и возможные режимы адресации.

Данные команды предназначены для выполнения операций «булевой» алгебры (И, ИЛИ, НЕ, Исключающее ИЛИ) над байтами и отдельными битами. Например, если аккумулятор содержит значение 00110101B и имеется константа <byte> со значением 01010011B, то после выполнения операции:

ANL A , <byte>

в аккумуляторе будет содержаться значение 00010001B.

В качестве примера использования режимов адресации рассмотрим команду ANL:

```
ANL  A, 7Fh      ; прямая адресация;
ANL  A, @R1      ; косвенная адресация;
ANL  A, R6       ; регистровая адресация;
ANL  A, #53h     ; непосредственная адресация.
```

Все логические операции над содержимым аккумулятора при тактовой частоте 12 МГц занимают 1 мкс, остальные – 2 мкс. Логические операции могут производиться над любым из нижних 128 байтов внутренней памяти данных или над любым регистром в пространстве SFR в режиме прямой адресации без использования аккумулятора. Так, например, для быстрого инвертирования содержимого порта P1, можно применить команду:

XLR P1, #0ffh

Операции циклического сдвига (RL A, RLC A и т.д.) перемещают содержимое аккумулятора на 1 бит вправо или влево. В случае левого циклического сдвига младший бит перемещается в старшую позицию.

Операция SWAP A осуществляет обмен младшей и старшей тетрад (полубайт) в двоичное число, меньшее, чем 100. Оно может быть быстро преобразовано в BCD-форму с помощью следующих действий:

```
MOV   B, #10
DIV   A B
SWAP  A
ADD   A, B
```

Деление числа на 10 пересылает количество «десятков» в младшую тетраду аккумулятора, а количество «единиц» – в регистр B. Инструкции SWAP и ADD переносят десятки в старший полубайт аккумулятора, в единицы – в младший.

Таблица 14

Логические инструкции MCS-51

Мнемоника	Действие	Режим адресации	Время вып., Мкс
ANL A, byte	A=A .and. byte	Dir, Ind, Reg, Imm	1
ANL byte, A	byte=byte .and. A	Dir	1
ANL byte, #data	byte=byte .and- #data	Dir	2
ORL A,byte	A=A .or.byte	Dir, Ind, Reg, Imm	1
ORL byte, A	byte=byte .or. A	Dir	1
ORL byte, #data	byte=byte .or. #data	Dir	2
XRL A, byte	A=A .xor. A	Dir, Ind, Reg, Imm	1
XRL byte, A	byte=byte .xor. A	Dir	1

XRL byte, #data	byte=byte .xor. #data	Dir	2
CLR A	A=00h	Аккумулятор	1
CPL A	A= .not. A	»	1
RL A	Сдвиг влево на 1 бит	»	1
RLC A	То же через С-бит	»	1
RR A	Сдвиг вправо на 1 бит	»	1
RRC A	То же через С-бит	»	1
SWAP A	Обменять местами полубайты	»	1

7.3.3. Команды передачи данных

Внутренняя память данных. В табл. 15 показаны команды, предназначенные для пересылок данных во внутреннем ОЗУ и применяемые при этом режиме адресации. При частоте 12 МГц все пересылки осуществляются за период в 1 или 2 мкс.

Команда **MOV <dest> , <src>** позволяет пересылать данные между ячейками внутреннего ОЗУ или SFR без использования аккумулятора. При этом работа с верхними 128 байтами внутреннего ОЗУ может осуществляться только в режиме косвенной адресации, а обращение к регистрам SFR – только в режиме прямой адресации.

Во всех микросхемах серии MCS-51 стек размещается непосредственно в резидентной памяти данных чипа и увеличивается вверх. Инструкции **PUSH** вначале увеличивают значение указателя стека (**SP**), а затем записывают в стек байт данных. Команды **PUSH** и **POP** используется только в режиме прямой адресации, записывая или восстанавливая байт. Стек является всегда доступным при косвенной адресации через регистр **SP**. Таким образом, стек может использовать и верхние 128 байт памяти данных, если они есть на кристалле. Эти же соображения исключают возможность использования стековых команд для адресации области SFR. В тех кристаллах, где верхняя 128-байтная половина памяти данных отсутствует, увеличение стека за границу 128 байт ведет к потере данных при записи и восстановлении (они в этом случае становятся неопределенными).

Инструкции передачи данных включают в себя 16-битовые операции пересылки **MOV**, которые могут использоваться для инициализации регистра указателя данных **DPTR**, при просмотре таблиц в программной памяти или для доступа к внешней памяти данных.

Операция **XCH A, <byte>** применяется для обмена данными между аккумулятором и адресуемым байтом. Команда **XCHD A, @Ri** аналогична предыдущей, но выполняется только для младших тетрад, участвующих в обмене.

Таблица 15

Инструкции передачи данных MCS-51, использующие внутреннюю память данных

Мнемоника	Действие	Режимы адресации	Время вып., мкс
1	2	3	4
MOV A , src	A=src	Dir, Ind, Reg, Imm	1
MOV dest , A	dest=A	Dir, Ind, Reg	1
MOV dest , src	dest=src	Dir, Ind, Reg, Imm	2

Окончание табл..15

1	2	3	4
MOV DPTR, #data16	DPTR=16-битная константа	Imm	2
PUSH src	INC SP; MOV @SP , src	Dir	2
POP dest	MOV dest , @SP; DEC SP	Dir	2
XCH A , byte	A<=>byte	Dir, Ind, Reg	1
XCHD A , @Ri	A<=>@Ri	Ind	1

Внешняя память данных. В табл. 16 показаны инструкции перехода, предназначенные для доступа к внешней памяти данных (указанное время выполнения действительно для частоты 12 МГц). При этом может использоваться только косвенная адресация. В случае однобайтных адресаций используется один из регистров: **R1** или **R0** текущего регистрового банка, а для 16-разрядных – регистр **DPTR**. Отрицательной стороной 16-разрядной адресации является то, что при небольшом объеме внешнего ОЗУ для передачи адреса используются все 8 бит порта **P2**. С другой стороны, 8-битная адресация в небольших системах позволяет не задействовать порт **P2**.

Сигналы чтения и записи (**RD** и **WR**) во внешнее ОЗУ активизируются только во время выполнения инструкций **MOVX.** Обычно эти сигналы неактивны, и если они не задействованы, то соответствующие им выводы можно использовать как дополнительные линии портов ввода-вывода.

Таблица 16

Инструкции передачи данных MCS-51, использующие внешнюю память данных

Адрес, бит	Мнемоника	Действие	Время вып., мкс
8	MOVX A, @Ri	A=@Ri	2
8	MOVX @Ri, A	@Ri=A	2
16	MOVX A, @DPTR	A=@DPTR	2
16	MOVX @DPTR, A	@DPTR=A	2

Операции с таблицами. В табл. 17 показаны 2 команды, предназначенные для чтения таблиц, размещенных в памяти программ. С помощью этих инструкций, осуществляющих доступ исключительно к памяти программ,

возможно только чтение таблиц, но не их изменение. Мнемонически эти команды выглядят как **MOVC** ("move constant").

Если таблица расположена во внешней программной памяти, то чтение байта из нее сопровождается стробом **PSEN.**

Таблица 17

Инструкции MCS-51 для работы с табличными наборами

Мнемоника	Действие	Время вып., мкс
MOVC A,@A+DPTR	A=@(A+DPTR)	2
MOVC A,@A+PC	A=@(A+PC)	2

Первая команда предназначена для обращения к таблице с максимальным числом входов до 256 (от 0 до 255). Номер требуемого ввода в таблицу загружается в аккумулятор, а регистр **DPTR** устанавливается на точку начала таблицы. Например:

MOVC A , @ A+DPTR

копирует содержимое требуемой позиции таблицы в аккумулятор.

Другая инструкция **MOVC** работает аналогичным образом за исключением того, что в качестве указателя базы используется счетчик команд – **PC** и обращение к таблице производится из подпрограммы. Вначале номер требуемой точки входа загружается в аккумулятор, затем вызывается подпрограмма:

```
MOV     A , Entry_Number
CALL    Table
```

Подпрограмма "Table" будет выглядеть следующим образом:

```
Table:
        MOVC    A, @A+PC
        RET
```

Сама таблица находится в памяти программ непосредственно за инструкцией **RET**. Такая таблица может иметь до 255 точек входа, пронумерованных от 1 до 255. Номер 0 не используется, потому что во время выполнения инструкции **MOVC,** счетчик команд **PC** содержит адрес инструкции **RET,** и значением точки входа 0 будет сам код этой инструкции.

7.3.4. Логические операции

Микросхемы MCS-51 содержат в своем составе «булевый» процессор. Внутреннее ОЗУ имеет 128 прямо-адресуемых бит. Точно так же пространство регистров SFR может поддерживать до 128 битовых полей. Побитно адресуемыми являются все порты ввода/вывода, каждая линия которых может рассматриваться как однобитовый порт. Инструкции, осуществляющие доступ к битам, представляют из себя не только условные переходы, но и пересылки, сброс, инверсии, операции «И» и «ИЛИ». Выполнение этого типа битовых операций на других архитектурах затруднительно как с точки зрения объема необходимого программного кода, так и времени выполнения.

Набор «булевских» инструкций перечислен в табл.18. Все биты доступны при прямой адресации. Битовые адреса с 00h по 7Fh расположены в нижней

128-байтной области внутреннего ОЗУ (Lower 128), а адреса с 80h по 7Fh – в пространстве регистров SFR.

Продемонстрируем, как осуществляется перенос внутреннего флага на вывод порта:

MOV C, FLAG
MOV P1.0, C

В этом примере FLAG – имя любого адресуемого бита в нижней 128-байтной области внутреннего ОЗУ или SFR. Линия порта ввода-вывода (в данном случае младший бит) устанавливается или очищается в зависимости от значения флажка (0 или 1).

Бит переноса (Carry Bit) в **PSW** используется как 1-битный аккумулятор булевого процессора. В отдельную группу входят битовые инструкции, ссылающиеся на этот бит как на «**C**» (как, например **CRL C**). Этот бит имеет и прямой адрес, поскольку **PSW** является побитно-адресуемым регистром, а бит **C** входит в его состав.

Таблица 18

Логические инструкции MCS-51

Мнемоника	Действие	Время вып., мкс
ANL C, Bit	C=C .and. bit	2
ANL C,/bit	C=C .and. (.not. bit)	2
ORL C,bit	C=C .or. bit	2
ORL C,/bit	C=C. or. (.not. bit)	2
MOV C, bit	C=bit	1
MOV bit C	bit=C	2
CLR C	C=0	1
CLR bit	bit=0	1
SETB C	C=1	1
SETB bit	bit=1	1
CPL C	C=.not. C	1
CPL bit	bit=.not. bit	1
JC rel	Переход, если C=1	2
JNC rel	» если C=0	2
JB bit, rel	» если bit=1	2
JNB bit, rel	» если bit=0	2
JBC bit, rel	» если bit=1; CLR bit	2

Отметим, что набор логических инструкций имеет в своем составе такие операции, как **ANL** и **ORL ,** операция же «исключающее ИЛИ» (**XRL**) реализуется программно. К примеру, требуется получить «исключающее ИЛИ» над двумя битами:

C = bit1 .XRL. bit2

Программа реализации будет выглядеть следующим образом:

MOV C , bit1

```
        JNB     bit2 , Over
        CPL     C
Over: . . .
```

Вначале **bit1** пересылается в бит переноса «**C**». Если **bit2**=0 , то «**C**» содержит правильный результат. То есть, **bit1 .XOR. bit2 = bit1** , если **bit2**=0. В противном случае, если **bit2**=1, «**C**» содержит инверсию результата, и для получения правильного значения его необходимо инвертировать (**CPL C**).

В этом фрагменте используется одна из инструкций тестирования битов **JNB**. Также имеются команды, выполняющие переход в том случае, если требуемый бит установлен (**JC, JB, JBC**) или не установлен (**JNC, JNB**). В рассмотренном выше примере тестируется **bit2**. Если он равен 0, то производится переход. **JBC** выполняет переход в случае, если требуемый бит установлен и «обнуляет» его. Таким образом, бит может быть проверен и очищен за одну операцию.

Все биты ССП (**PSW**) прямо-адресуемы, а биты четности и флаги пользователя доступны и для команд, проверяющих биты.

Адреса вышеперечисленных операций переходов обозначаются на языке ассемблера меткой либо реальным значением в пространстве памяти программ. Адреса условных переходов ассемблируются в относи-тельное смещение – знаковый (дополненный до двух) байт, прибавляемый к программному счетчику (**PC**) в случае выполнения условия перехода.

Границы таких переходов лежат в пределах между -128 и +127 байт, относительно первого байта следующего за инструкцией.

7.3.5. Инструкции переходов

В табл. 19 дан список безусловных переходов. Однако показана только одна команда «**JMP addr**», хотя, фактически, их имеется 3 варианта: **SJMP, LJMP, AJMP**, различающиеся форматом адреса назначения. Мнемоника **JMP** используется в том случае, когда программист не уверен, какой вариант необходимо применить и предоставляет выбор Ассемблеру.

Инструкция **SJMP** кодирует адрес как относительное смещение, и занимает 2 байта – код операции и относительное смещение в 1 байт. Дальность перехода ограничена диапазоном -128/+127 байт относительно инструкции, следующей за **SJMP**.

Таблица 19

Инструкции безусловных переходов MCS-51

Мнемоника	Действие	Время вып. мкс
JMP addr	Переход по адресу PC=addr	2
JMP @A+DPTR	Переход по вычисляемому адресу PC=A+DPTR	2
CALL addr	Вызов подпрограммы	2
RET	Возврат из подпрограммы	2

RETI	» » обработчика прерывания	2
NOP	Пустая команда	1

В инструкции **LJMP** используется адрес назначения в виде 16-битной константы. Длина команды составляет 3 байта: 1 байт для кода операции и 2 байта адреса. Адрес назначения может располагаться в любом месте 64К-байтного пространства памяти программ.

AJMP использует 11-битную константу адреса. Команда состоит из двух байт: кода операции, содержащего 3 старших бита 11-битового адреса, и младшего байта адреса перехода. При выполнении инструкции младшие 11 бит адресного счетчика замещаются 11-битным адресом команды. Пять старших бит **PC** остаются неизменными. Таким образом, переход может производиться внутри 2К-байтного блока, в котором располагается инструкция, следующая за командой **AJMP**.

Во всех случаях программист определяет адрес назначения на Ассемблере двумя способами: как метку, либо как 16-битную константу. Ассемблер будет помещать адрес назначения в команде в правильном формате. Если формат инструкции не поддерживает дистанцию до адреса назначения, то в листинге выдается сообщение об ошибке: «**Destination out of range**».

Команда **JMP @A+DPTR** образует переход «по выбору». Адрес назначения вычисляется как сумма 16-битного значения регистра **DPTR** и аккумулятора. Обычно **DPTR** устанавливается на адрес таблицы переходов, а аккумулятор содержит индекс этой таблицы. Например, для того чтобы осуществить ветвление с 5 вариантами выбора (от 0 до 4), необходимо выполнить следующий фрагмент:

```
MOV   DPTR , #Jump_Table
MOV   A , Index_Number
RL    A
JMP   @A+DPTR
```

Инструкция **RL A** преобразует индекс (значение от 0 до 4) в четное число от 0 до 8, поскольку каждая точка входа в таблице переходов занимает 2 байта:

```
Jump_Table:
    AJMP   CASE_0
    AJMP   CASE_1
    AJMP   CASE_2
    AJMP   CASE_3
    AJMP   CASE_4
```

В табл. 19 показана одна инструкция «**CALL addr**», хотя в действительности их две: **LCALL** и **ACALL** . Эти команды различаются представлением адреса подпрограммы. Мнемоника **CALL** используется программистом в случаях, когда нет уверенности, как расположен адрес.

Инструкция **LCALL** использует 16-битный адресный формат – подпрограмма может быть расположена в любом месте 64Кбайт памяти программ. **ACALL** применяет 11-битный формат - подпрограмма должна находиться в одном 2 Кбайтном блоке с инструкцией, следующей за **ACALL**.

В любом случае программист определяет адрес подпрограммы на Ассемблере либо как метку, либо как 16-битную константу. Ассемблер помещает этот адрес в инструкцию в определенном формате.

Подпрограмма завершается инструкцией **RET**, позволяющей вернуться на инструкцию, следующую за командой **CALL**.

Инструкция **RETI** используется для возврата из обработчиков прерываний. Единственное различие между **RET** и **RETI** состоит в том, что **RETI** информирует управляющую систему о том, что обработка прерывания завершилась. Если в момент выполнения инструкции **RETI** нет других прерываний, то она идентична **RET**.

Таблица 20

Инструкции условных переходов MCS-51

Мнемоника	Действие	Режимы адресации	Время вып., мкс
JZ rel	Переход, если A=0	Аккумулятор	2
JNZ rel	» если A<>0	»	2
DJNZ byte, rel	DEC byte; переход, если не равно 0	Dir, Reg	2
CJNE A, byte, rel	Переход, если A<>byte	Dir, Imm	2
CJNE byte, #data, rel	» если byte<>data	Ind, Reg	2

В табл. 20 показаны команды условных переходов. Все они определяют адрес назначения как относительное смещение, ограниченное длиной перехода в -128/+127 байт от инструкции, следующей за условным переходом. Важно отметить, что пользователь может определить адрес на Ассемблере такими же методами, как и для других переходов, т.е. меткой и 16-битной константой.

В ССП (**PSW**) отсутствует флажок нуля, поэтому инструкции **JZ** и **JNZ** проверяют условие «равен нулю» тестированием данных в аккумуляторе.

Инструкция **DJNZ (**Decrement and Jump if Not - «уменьшить и перейти, если не равно»**)** предназначена для управления циклами. Для выполнения цикла N раз надо загрузить в счетчик байт со значением N и закрыть тело цикла командой **DJNZ**, указывающей на начало цикла, как показано в следующем примере (для N=10):

```
MOV   R1, #10
Loop:                    ; начало цикла
. . .                    ; тело цикла
DJNZ  R1, Loop
. . .                    ; продолжение
```

Инструкция **CJNE** (Compare and Jump if Not Equal - «сравнить и перейти, если не равно») тоже может использоваться для организации циклов. В поле операндов инструкции имеется 2 байта. Переход производится только в случае, если эти байты не равны. Другим применением данной инструкции является проверка условий «**больше чем**», «**меньше чем**». Два байта в поле операндов представлены как «беззнаковое целое». Если первый операнд меньше, чем второй, то бит переноса «**C**» устанавливается в 1.

Контрольные вопросы

1. Дайте общую характеристику директив языка Ассемблер.
2. Сформулируйте назначение и приведите примеры использования директив ORG, EQU, SET.
3. Сформулируйте назначение и приведите примеры использования директив DATA, XDATA, BIT.
4. Сформулируйте назначение и приведите примеры использования директив DB, DS, DW.
5. Дайте характеристику директив условной трансляции.
6. Дайте характеристику режимов адресации. Приведите примеры.
7. Дайте характеристику арифметических и логических команд. Приведите примеры.
8. Дайте характеристику команд передачи данных. Приведите примеры.
9. Дайте характеристику команд логических операций. Приведите примеры.
10. Дайте характеристику команд переходов. Приведите примеры.