А.Е. БОЯРИНОВ, И.А. ДЬЯКОВ

АРХИТЕКТУРА МИКРОКОНТРОЛЛЕРОВ СЕМЕЙСТВА MCS-51

• ИЗДАТЕЛЬСТВО ТГТУ •

Министерство образования и науки Российской Федерации

Государственное образовательное учреждение
высшего профессионального образования
"Тамбовский государственный технический университет"

А.Е. Бояринов, И.А. Дьяков

АРХИТЕКТУРА МИКРОКОНТРОЛЛЕРОВ СЕМЕЙСТВА MCS-51

Конспект лекций для студентов всех форм обучения
специальностей 072000; 210200; 230104

Тамбов
Издательство ТГТУ
2005

УДК 681.3.06(07)
ББК □973.26-04я73-2
Б869

Р е ц е н з е н т

Кандидат технических наук, доцент
Н.Г. Чернышов

Бояринов, А.Е.
Б869 Архитектура микроконтроллеров семейства MCS-51 : конспект лекций / А.Е. Бояринов, И.А. Дьяков. Тамбов : Изд-во Тамб. гос. техн. ун-та, 2005. 64 с.

Рассмотрены схемотехника и программирование однокристальных микроЭВМ семейства MCS-51.

Предназначены для студентов всех форм обучения специальностей 072000, 210200, 230104.

УДК 681.3.06(07)
ББК □973.26-04я73-2

Учебное издание

БОЯРИНОВ Алексей Евгеньевич,
ДЬЯКОВ Игорь Алексеевич

АРХИТЕКТУРА МИКРОКОНТРОЛЛЕРОВ СЕМЕЙСТВА MCS-51

Конспект лекций

Редактор Т.М. Глинкина

Инженер по компьютерному макетированию М.Н. Рыжкова

Подписано к печати 23.05.2005.
Формат 60 × 84/16. Бумага офсетная. Печать офсетная.
Гарнитура Times New Roman. Объем: 3,72 усл. печ. л.; 3,6 уч.-изд. л.
Тираж 50 экз. С. 370^{M}

Издательско-полиграфический центр
Тамбовского государственного технического университета
392000, Тамбов, Советская, 106, к. 14

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

С 80-х годов XX века в микропроцессорной технике выделился самостоятельный класс интегральных схем – однокристальные микроконтроллеры, которые предназначены для встраивания в приборы различного назначения. От класса однокристальных микропроцессоров их отличает наличие внутренней памяти, развитые средства взаимодействия с внешними устройствами.

Широкое распространение получили 8-разрядные однокристальные микроконтроллеры семейства MCS-51. Это семейство образовалось на основе микроконтроллера Intel 8051, получившего большую популярность у разработчиков микропроцессорных систем контроля благодаря удачно спроектированной архитектуре. Архитектура микроконтроллера – это совокупность внутренних и внешних программно-доступных аппаратных ресурсов и системы команд.

Впоследствии фирма Intel выпустила около 50 моделей на базе операционного ядра микроконтроллера Intel 8051. Одновременно многие другие фирмы, такие как Atmel, Philips, начали производство своих микроконтроллеров, разработанных в стандарте MCS-51. Существует также и отечественный аналог микроконтроллера Intel 8051 – микросхема К1816ВЕ51.

1 СТРУКТУРА МИКРОКОНТРОЛЛЕРА INTEL 8051

Микроконтроллер Intel 8051 выполнен на основе высокоуровневой *n*-МОП технологии. Его основные характеристики следующие:

- восьмиразрядный центральный процессор, оптимизированный для реализации функций управления;
- встроенный тактовый генератор (максимальная частота 12 МГц);
- адресное пространство памяти программ – 64 Кбайт;
- адресное пространство памяти данных – 64 Кбайт;
- внутренняя память программ – 4 Кбайт;
- внутренняя память данных – 128 байт;
- дополнительные возможности по выполнению операций булевой алгебры (побитовые операции);
- 32 двунаправленные и индивидуально адресуемые линии ввода/вывода;
- два 16-разрядных многофункциональных таймера/счетчика;
- полнодуплексный асинхронный приемопередатчик (последовательный порт);
- векторная система прерываний с двумя уровнями приоритета и пятью источниками событий.

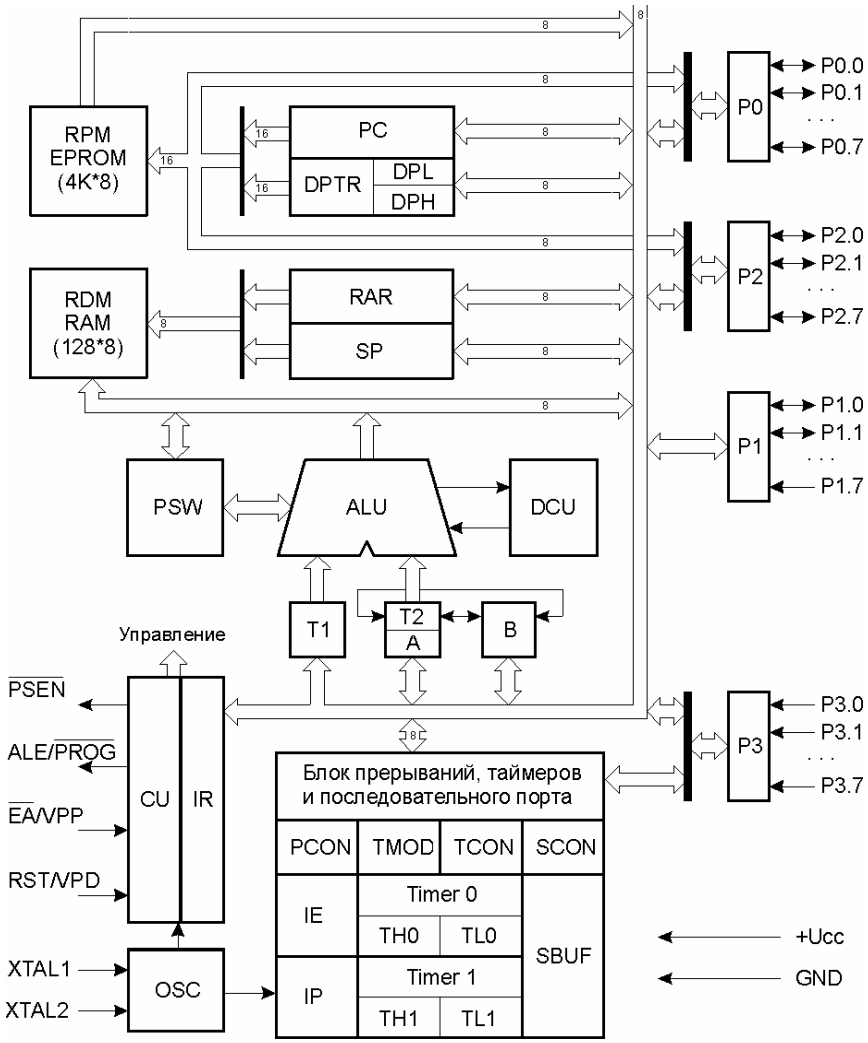


Рис. 1 Структурная схема микроконтроллера Intel 8051

Основу структурной схемы микроконтроллера (рис. 1) образует внутренняя двунаправленная 8-разрядная шина, которая связывает между собой основные узлы и устройства микроконтроллера: резидентную память программ (RPM), резидентную память данных (RDM), арифметико-логическое устройство (ALU), блок регистров специальных функций, устройство управления (CU), параллельные порты ввода/вывода (P0 – P3), а также программируемые таймеры и последовательный порт.

1.1 ОРГАНИЗАЦИЯ ПАМЯТИ

Данный микроконтроллер имеет встроенную (резидентную) и внешнюю память программ и данных. Резидентная память программ (RPM) имеет объем 4 Кбайт, резидентная память данных (RDM) – 128 байт.

В зависимости от модификации микроконтроллера RPM выполняется в виде масочного ПЗУ, однократно программируемого либо репрограммируемого ПЗУ.

При необходимости пользователь может расширять память программ установкой внешнего ПЗУ. Доступ к внутреннему или внешнему ПЗУ определяется значением сигнала на выводе EA (External Access):

EA = V_{CC} (напряжение питания) – доступ к внутреннему ПЗУ;

EA = V_{SS} (потенциал земли) – доступ к внешнему ПЗУ.

Внешняя память программ и данных может составлять по 64 Кбайт и адресоваться с помощью портов P0 и P2. На рис. 2 представлена карта памяти микроконтроллера Intel 8051.

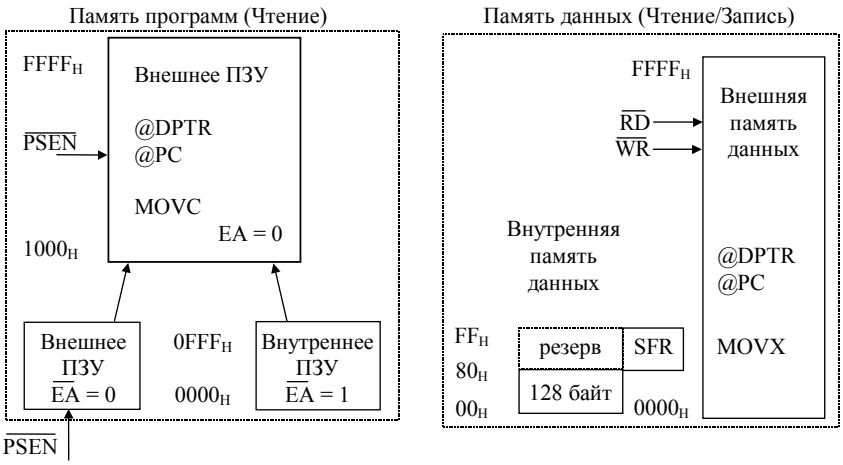


Рис. 2 Организация памяти Intel 8051

Строб чтения внешнего ПЗУ – $\overline{PSEN}$ (Program Store Enable) генерируется при обращении к внешней памяти программ и является неактивным во время обращения к ПЗУ, расположенному на кристалле.

Область нижних адресов памяти программ (рис. 3) используется системой прерываний. Архитектура микросхемы INTEL 8051 обеспечивает поддержку пяти источников прерываний. Адреса, по которым передается управление по прерыванию, называются векторами прерывания.

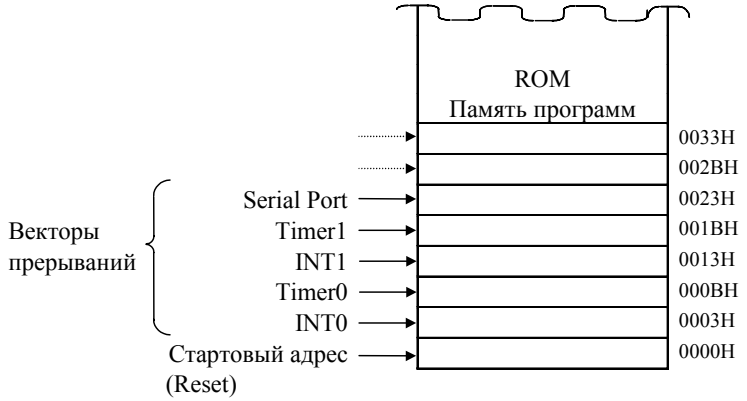


Рис. 3 Карта нижней области программной памяти

1.2 АРИФМЕТИКО-ЛОГИЧЕСКОЕ УСТРОЙСТВО

8-битное арифметико-логическое устройство (ALU) 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. К входам подключены программно недоступные регистры T1 и T2, предназначенные для временного хранения операндов, схема десятичной коррекции (DCU) и схема формирования признаков результата операции (PSW).

Простейшая операция сложения используется в ALU для инкрементирования содержимого регистров, продвижения регистра-указателя данных (RAR) и автоматического вычисления следующего адреса резидентной памяти программ. Простейшая операция вычитания используется в ALU для декрементирования регистров и сравнения переменных.

Простейшие операции автоматически образуют "тандемы" для выполнения таких операций, как, например, инкрементирование 16-битных регистровых пар. В ALU реализуется механизм каскадного выполнения простейших операций для реализации сложных команд. Так, например, при выполнении одной из команд условной передачи управления по результату сравнения в ALU трижды инкрементируется счетчик команд (PC), дважды производится чтение из RDM, 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 всего лишь за 2 мкс.

Важной особенностью ALU является его способность оперировать не только байтами, но и битами. Отдельные программно-доступные биты могут быть установлены, сброшены, инвертированы, переда-

ны, проверены и использованы в логических операциях. Эта способность достаточно важна, поскольку для управления объектами часто применяются алгоритмы, содержащие операции над входными и выходными булевыми переменными, реализация которых средствами обычных микропроцессоров сопряжена с определенными трудностями.

Таким образом, ALU может оперировать четырьмя типами информационных объектов: булевыми (1 бит), цифровыми (4 бита), байтными (8 бит) и адресными (16 бит). В ALU выполняется 51 различная операция пересылки или преобразования этих данных. Так как используются 11 режимов адресации (7 – для данных и 4 – для адресов), то путем комбинирования операции и режима адресации базовое число команд 111 расширяется до 255 из 256 возможных при однобайтном коде операции.

1.3 РЕЗИДЕНТНАЯ ПАМЯТЬ ПРОГРАММ И ДАННЫХ

Резидентные (размещенные на кристалле) память программ (RPM) и память данных (RDM) физически и логически разделены, имеют различные механизмы адресации, работают под управлением различных сигналов и выполняют разные функции.

Память программ RPM имеет емкость 4 Кбайта и предназначена для хранения команд, констант, управляющих слов инициализации, таблиц перекодировки входных и выходных переменных и т.п. Память имеет 16-битную шину адреса, через которую обеспечивается доступ из программного счетчика PC или из регистра-указателя данных (DPTR). DPTR выполняет функции базового регистра при косвенных переходах по программе или используется в операциях с таблицами.

Память данных RDM предназначена для хранения переменных в процессе выполнения прикладной программы, адресуется одним байтом и имеет емкость 128 байт. Кроме того, к ее адресному пространству примыкают адреса регистров специальных функций, которые перечислены в табл. 1.

1 Блок регистров специальных функций

Символ		Наименование		Адрес
*	A	Аккумулятор		0E0H
*	B	Регистр-расширитель аккумулятора		0F0H
*	PSW	Слово состояния программы		0D0H
	SP	Регистр-указатель стека		81H
	DPTR	Регистр-указатель данных	(DPH)	83H
			(DPL)	82H

Продолжение табл. 1

Символ		Наименование	Адрес
*	P0	Порт 0	80H
*	P1	Порт 1	90H
*	P2	Порт 2	0A0H
*	P3	Порт 3	0B0H
*	IP	Регистр приоритетов прерываний	0B8H
*	IE	Регистр маски прерываний	0A8H
	TMOD	Регистр режима таймера/счетчика	89H
*	TCON	Регистр управления/статуса таймера	88H
	TH0	Таймер 0 (старший байт)	8CH
	TL0	Таймер 0 (младший байт)	8AH
	TH1	Таймер 1 (старший байт)	8DH
	TL1	Таймер 1 (младший байт)	8BH
*	SCON	Регистр управления приемопередатчиком	98H

SBUF	Буфер приемопередатчика	99H
PCON	Регистр управления мощностью	87H

П р и м е ч а н и е. Регистры, имена которых отмечены знаком (*), допускают адресацию отдельных битов.

Память программ, так же как и память данных, может быть расширена до 64 Кбайт путем подключения внешних микросхем.

1.4 АККУМУЛЯТОР И РЕГИСТРЫ ОБЩЕГО НАЗНАЧЕНИЯ

Аккумулятор (А) является источником операнда и местом фиксации результата при выполнении арифметических, логических операций и ряда операций передачи данных. Кроме того, только с использованием аккумулятора могут быть выполнены операции сдвигов, проверка на нуль, формирование флага паритета и т.п.

В распоряжении пользователя имеются четыре банка по 8 регистров общего назначения R0 – R7 (рис. 9). Однако возможно использование регистров только одного из четырех банков, который выбирается с помощью бит регистра PSW.

1.5 РЕГИСТР СЛОВА СОСТОЯНИЯ ПРОГРАММЫ И ЕГО ФЛАГИ

При выполнении многих команд в ALU формируется ряд признаков операции (флагов), которые фиксируются в регистре слова состояния программы (PSW). В табл. 2 приводится перечень флагов PSW, даются их символические имена и описываются условия их формирования.

2 Формат слова состояния программы PSW

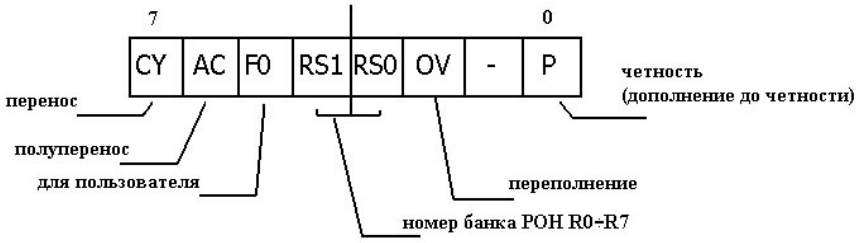
Символ	Разряд	Имя и назначение
CY	PSW.7	Флаг переноса. Устанавливается и сбрасывается аппаратно или программно при выполнении арифметических и логических операций
AC	PSW.6	Флаг вспомогательного переноса. Устанавливается и сбрасывается только аппаратно при выполнении команд сложения и вычитания и сигнализирует о переносе или займе 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 программно для выбора рабочего банка регистров (табл. 3)
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 и фиксирует нечетное/четное число единичных битов в аккумуляторе, т.е. выполняет контроль по четности

3 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	18H – 1FH

Наиболее "активным" флагом PSW является флаг переноса, который принимает участие и модифицируется в процессе выполнения множества операций, включая сложение, вычитание и сдвиги. Кроме того, флаг переноса (CY) выполняет функции "булева аккумулятора" в командах, манипулирующих с битами. Флаг переполнения (OV) фиксирует арифметическое переполнение при операциях над целыми числами со знаком и делает возможным использование арифметики в дополнительных кодах. ALU не управляет флагами селекции банка регистров (RS0, RS1), их значение полностью определяется прикладной программой и используется для выбора одного из четырех регистровых банков.

В виде байта регистр PSW может быть представлен следующим образом:



В микропроцессорах, архитектура которых опирается на аккумулятор, большинство команд работают с ним, используя неявную адресацию. В Intel 8051 дело обстоит иначе. Хотя процессор имеет в своей основе аккумулятор, он может выполнять множество команд и без его участия. Например, данные могут быть переданы из любой ячейки RDM 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 без использования аккумулятора. Флаги и управляющие биты могут быть проверены и изменены аналогично.

1.6 РЕГИСТРЫ-УКАЗАТЕЛИ

8-битный указатель стека (SP) может адресовать любую область RDM. Его содержимое инкрементируется прежде, чем данные будут запомнены в стеке в ходе выполнения команд PUSH и CALL. Содержимое SP декрементируется после выполнения команд POP и RET. Подобный способ адресации элементов стека называют прединкрементным/постдекрементным. В процессе инициализации микроконтроллера после сигнала RST в SP автоматически загружается код 07H. Это значит, что если прикладная программа не переопределяет стек, то первый элемент данных в стеке будет располагаться в ячейке RDM с адресом 08H.

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

1.7 РЕГИСТРЫ СПЕЦИАЛЬНЫХ ФУНКЦИЙ

Регистры с символическими именами IP, IE, TMOD, TCON, SCON и PCON используются для фиксации и программного изменения управляющих бит и бит состояния схемы прерывания, таймера/счетчика, приемопередатчика последовательного порта и для управления энергопотреблением. Под-

робно их организация будет описана в разделах 1.8 – 1.12 при рассмотрении особенностей работы микроконтроллера в различных режимах.

1.8 УСТРОЙСТВО УПРАВЛЕНИЯ И СИНХРОНИЗАЦИИ

Кварцевый резонатор, подключаемый к внешним выводам микроконтроллера, управляет работой внутреннего генератора, который в свою очередь формирует сигналы синхронизации. Устройство управления (CU) на основе сигналов синхронизации формирует машинный цикл фиксированной длительности, равной 12 периодам генератора. Большинство команд микроконтроллера выполняется за один машинный цикл. Некоторые команды, оперирующие с 2-байтными словами или связанные с обращением к внешней памяти, выполняются за два машинных цикла. Только команды деления и умножения требуют четырех машинных циклов. На основе этих особенностей работы устройства управления производится расчет времени исполнения прикладных программ.

На схеме микроконтроллера к устройству управления примыкает регистр команд (IR). В его функцию входит хранение кода выполняемой команды.

Входные и выходные сигналы устройства управления и синхронизации:

1 PSEN – разрешение программной памяти;
2 ALE – выходной сигнал разрешения фиксации адреса;
3 PROG – сигнал программирования;
4 EA – блокировка работы с внутренней памятью;
5 VPP – напряжение программирования;
6 RST – сигнал общего сброса;
7 VPD – вывод резервного питания памяти от внешнего источника;
8 XTAL – входы подключения кварцевого резонатора.

1.9 ПАРАЛЛЕЛЬНЫЕ ПОРТЫ ВВОДА/ВЫВОДА ИНФОРМАЦИИ

Все четыре порта (P0 – P3) предназначены для ввода или вывода информации побайтно. Каждый порт содержит управляемые регистр-защелку, входной буфер и выходной драйвер.

Выходные драйверы портов P0 и P2, а также входной буфер порта P0 используются при обращении к внешней памяти. При этом через порт P0 в режиме временного мультиплексирования сначала выводится младший байт адреса, а затем выдается или принимается байт данных. Через порт P2 выводится старший байт адреса в тех случаях, когда разрядность адреса равна 16 бит.

Все выводы порта P3 могут быть использованы для реализации альтернативных функций, перечисленных в табл. 4. Эти функции могут быть задействованы путем записи 1 в соответствующие биты регистра-защелки (P3.0 – P3.7) порта P3.

4 Альтернативные функции порта P3

Символ	Разряд	Имя и назначение
RD	P3.7	Чтение. Активный сигнал низкого уровня формируется аппаратно при обращении к внешней памяти данных
WR	P3.6	Запись. Активный сигнал низкого уровня формируется аппаратно при обращении к внешней памяти данных
T1	P3.5	Вход таймера/счетчика 1 или тест-вход
T0	P3.4	Вход таймера/счетчика 0 или тест-вход
INT1	P3.3	Вход запроса прерывания 1. Воспринимается сигнал низкого уровня или срез
INT0	P3.2	Вход запроса прерывания 0. Воспринимается сигнал низкого уровня или срез

TXD	P3.1	Выход передатчика последовательного порта в режиме UART. Выход синхронизации в режиме регистра сдвига
RXD	P3.0	Вход приемника последовательного порта в режиме UART. Ввод/вывод данных в режиме регистра сдвига

Порт 0 является двунаправленным, а порты 1 – 3 – квазидвунаправленными. Каждая линия портов может быть использована независимо для ввода или вывода.

По сигналу RST 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 P0 и P2 не могут быть использованы для этой цели в случае, если система имеет внешнюю память, связь с которой организуется через общую разделяемую шину адреса/данных, работающую в режиме временного мультиплексирования.

Обращение к портам ввода/вывода возможно с использованием команд, оперирующих с байтом, отдельным битом, произвольной комбинацией битов. При этом в тех случаях, когда порт является одновременно операндом и местом назначения результата, устройство управления автоматически реализует специальный режим, который называется "чтение –модификация – запись". Этот режим обращения предполагает ввод сигналов не с внешних выводов порта, а из его регистра-защелки, что позволяет исключить неправильное считывание ранее выведенной информации. Этот механизм обращения к портам реализован в командах:

1 ANL – логическое И, например, ANL P1, A;
2 ORL – логическое ИЛИ, например, ORL P2, A;
3 XRL – исключающее ИЛИ, например, XRL P3, A;
4 JBC – переход, если в адресуемом бите единица, и последующий сброс бита, например, JBC P1.1, LABEL;
5 CPL – инверсия бита, например, CPL P3.3;
6 INC – инкремент порта, например, INC P2;
7 DEC – декремент порта, например, DEC P2;
8 DJNZ – декремент порта и переход, если его содержимое не равно нулю, например, DJNZ r, LABEL;
9 MOV PX.Y,C – передача бита переноса в бит Y порта X;
10 SET PX.Y – установка бита Y порта X;
11 CLR PX.Y – сброс бита Y порта X.

1.10 ТАЙМЕРЫ/СЧЕТЧИКИ

В составе микроконтроллера имеются регистровые пары с символическими именами TH0, TL0 и TH1, TL1, на основе которых функционируют два независимых программно-управляемых 16-битных таймера/счетчика событий (T/C0 и T/C1). При работе в качестве таймера содержимое T/C инкрементируется в каждом машинном цикле, т.е. через каждые 12 периодов резонатора. При работе в качестве счетчика содержимое T/C инкрементируется под воздействием перехода из 1 в 0 внешнего входного сигнала, подаваемого на соответствующий (T0, T1) вход микроконтроллера. Опрос сигналов выполняется в каждом машинном цикле. Так как на распознавание перехода требуется два машинных цикла, то максимальная частота подсчета входных сигналов равна 1/24 частоты резонатора. На длительность периода входных сигналов ограничений сверху нет. Для гарантированного прочтения входного считываемого сигнала он должен удерживать значение 1 как минимум в течение одного машинного цикла.

Использование таймеров возможно в четырех режимах. Для управления режимами работы и для организации взаимодействия таймеров с системой прерывания используются два регистра специальных функций TMOD и TCON, описание которых приводится в табл. 5 – 7. Для обоих T/C режимы работы 0, 1 и 2 одинаковы. Режимы 3 для T/C0 и T/C1 различны.

5 Регистр режима работы таймера/счетчика

Символ	Разряд	Имя и назначение
GATE	TMOD.7 для T/C1 TMOD.3 для T/C0	Управление блокировкой. Если бит установлен, то таймер/счетчик x разрешен до тех пор, пока на входе INT x высокий уровень и бит управления TRx установлен. Если бит сброшен, то T/C разрешается, как только бит управления TRx устанавливается
C/T	TMOD.6 для T/C1 TMOD.2 для T/C0	Бит выбора режима таймера или счетчика событий. Если бит сброшен, то работает таймер от внутреннего источника сигналов синхронизации. Если бит установлен, то работает счетчик от внешних сигналов на входе Tx
M1	TMOD.5 для T/C1 TMOD.1 для T/C0	Режим работы (см. табл. 6)
M0	TMOD.4 для T/C1 TMOD.0 для T/C0	

6 Режимы работы таймера/счетчика

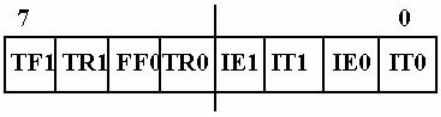
M1	M0	Режим работы
0	0	TLx работает как 5-битный предделитель
0	1	16-битный таймер/счетчик. THx и TLx включены последовательно
1	0	8-битный автоперезагружаемый таймер/счетчик. THx хранит значение, которое должно быть перезагружено в TLx каждый раз по переполнению
1	1	Таймер/счетчик 1 останавливается. Таймер/счетчик 0: TL0 работает как 8-битный таймер/счетчик, и его режим определяется управляющими битами таймера 0. TH0 работает только как 8-битный таймер, и его режим определяется управляющими битами таймера 1

7 Регистр управления/статуса таймера

Символ	Разряд	Имя и назначение
TF1	TCON.7	Флаг переполнения таймера 1. Устанавливается аппаратно при переполнении таймера/счетчика. Сбрасывается при обслуживании прерывания аппаратно

TR1	TCON.6	Бит управления таймера 1. Устанавливается/сбрасывается программой для пуска/останова
TF0	TCON.5	Флаг переполнения таймера 0. Устанавливается аппаратно. Сбрасывается при обслуживании прерывания
TR0	TCON.4	Бит управления таймера 0. Устанавливается/сбрасывается программой для пуска/останова таймера/счетчика
IE1	TCON.3	Флаг фронта прерывания 1. Устанавливается аппаратно, когда детектируется срез внешнего сигнала INT1. Сбрасывается при обслуживании прерывания
IT1	TCON.2	Бит управления типом прерывания 1. Устанавливается/сбрасывается программно для спецификации запроса INT1 (срез/низкий уровень)
IE0	TCON.1	Флаг фронта прерывания 0. Устанавливается по срезу сигнала INT0. Сбрасывается при обслуживании прерывания
IT0	TCON.0	Бит управления типом прерывания 0. Устанавливается/сбрасывается программно для спецификации запроса INT0 (срез/низкий уровень)

В виде байта регистр TCON можно изобразить следующим образом:



Режим 0. Перевод любого Т/С в этот режим делает его 8-разрядным таймером, на вход которого подключен делитель частоты на 32. В этом режиме таймерный регистр имеет разрядность 13 бит, в котором регистр THx работает как 8-разрядный счетчик, а регистр TLx как 5-битный предварительный делитель.

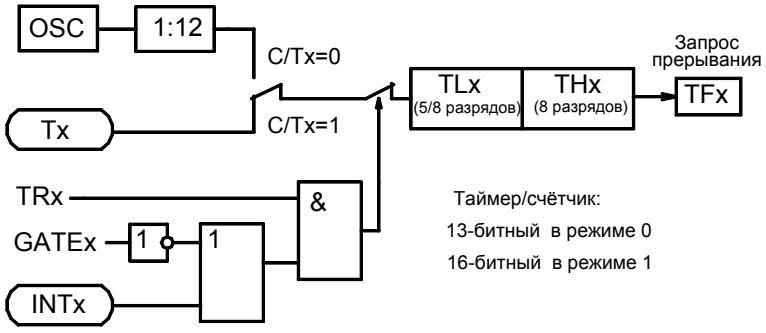


Рис. 4 Функциональная схема таймера в режимах 0 и 1

Поясним работу на примере таймера 1 (рис. 4). Входной синхросигнал разрешен (поступает на вход таймерного регистра), когда управляющий бит TR1 установлен в 1 и либо управляющий бит GATE (блокировка) равен 0, либо на внешний вход запроса прерывания INT1 поступает уровень 1.

В зависимости от состояния бита C/T1 счетчик будет считать либо внутреннюю частоту (режим таймера), либо внешние импульсы на выводе T1 (режим счетчика). Бит GATE1 позволяет использовать режим таймера для измерения длительности импульсного сигнала, подаваемого на вход внешнего прерывания INT1.

При переполнении счетчика TH1 (переход из состояния "все единицы" в состояние "все нули") устанавливается флаг прерывания от таймера TF1.

Режим 1. Работа любого T/C (рис. 4) в этом режиме такая же, как и в режиме 0, за исключением того, что таймерный регистр имеет разрядность 16 бит. Регистр THx является старшим 8-разрядным счетчиком, а TLx – младшим.

Режим 2 называется 8-битный с перезагрузкой (рис. 5). Переполнение 8-битного счетчика TLx приводит не только к установке флага TFx, но и

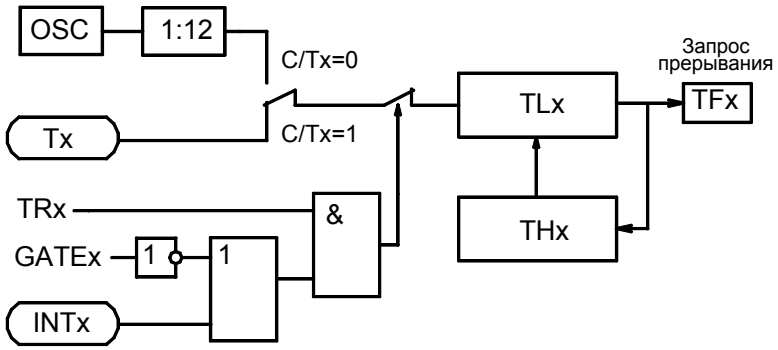


Рис. 5 Функциональная схема таймера в режиме 2

автоматически перезагружает в TLx содержимое старшего регистра THx. Перезагрузка оставляет содержимое THx неизменным.

Значение THx должно быть предварительно задано программным путем. Это позволяет формировать временные интервалы заданной длительности. В режиме 2 T/C0 и T/C1 работают одинаково, но T/C1 может использоваться для задания скорости работы последовательного порта.

Режим 3. В этом режиме (рис. 6) T/C0 и T/C1 работают по-разному. T/C1 сохраняет неизменным свое текущее содержимое. Иными словами, эффект такой же, как и при сбросе управляющего бита TR1 в нуль. В этом режиме TL0 и TH0 функционируют как два независимых 8-битных счетчика. Работу TL0 определяют управляющие биты T/C0 (C/T, GATE, TR0), входной сигнал INT0 и флаг переполнения TF0. Работу TH0, который может выполнять только функции таймера (подсчет машинных циклов микроконтроллера), определяет управляющий бит TR1. При этом TH0 использует флаг переполнения TF1.

Режим 3 используется в тех случаях, когда требуется наличие дополнительного 8-битного таймера или счетчика событий. Можно считать, что в режиме 3 микроконтроллер имеет в своем составе три таймера/счетчика. В случае, если T/C0 используется в режиме 3, T/C1 может быть или включен, или выключен, или переведен в свой собственный режим 3, или может быть использован последовательным портом в качестве генератора частоты передачи, или, наконец, может быть использован в любом применении, не требующем прерывания.

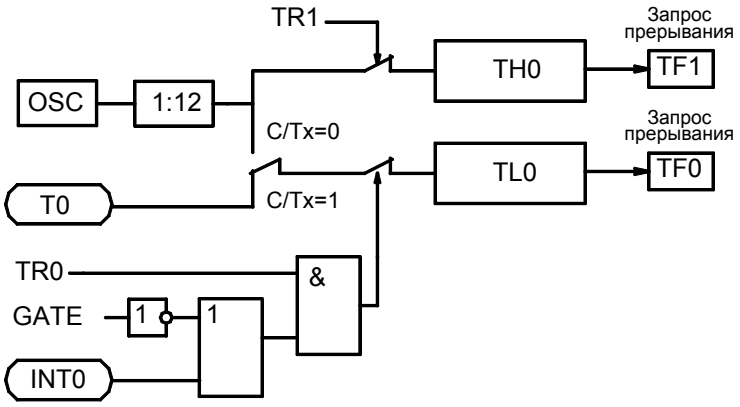


Рис. 6 Функциональная схема таймера в режиме 3

1.11 ПОСЛЕДОВАТЕЛЬНЫЙ ПОРТ

Через универсальный асинхронный приемопередатчик UART (Universal Asynchronous Receiver-Transmitter) происходят прием и передача информации, представленной последовательным кодом (младшими битами вперед) в полном дуплексном режиме обмена. В состав UART, называемого часто последовательным портом, входят принимающий и передающий сдвигающие регистры, а также специальный буферный регистр (SBUF) приемопередатчика.

1.11.1 Регистр SBUF

Регистр 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. 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 байта. Если к моменту окончания приема байта предыдущий байт не был считан, то он будет потерян.

1.11.2 Режимы работы последовательного порта

Последовательный порт может работать в четырех различных режимах.

Режим 0. Синхронный 8-битный режим с фиксированной скоростью. Информация передается и принимается через вход приемника RxD. Принимаются и передаются 8 бит данных. Через внешний выход передатчика TxD выдаются импульсы сдвига, которые сопровождают каждый бит. Частота передачи равна 1/12 частоты резонатора.

Режим 1. Асинхронный 8-битный режим с переменной скоростью. Через TxD передаются или из RxD принимаются 10 бит: старт-бит (0), 8 бит данных и стоп-бит (1). Скорость приема/передачи – величина переменная и задается таймером.

Режим 2. Асинхронный 9-битный режим с фиксированной скоростью. Через TxD передаются или из RxD принимаются 11 бит: старт-бит, 8 бит данных, программируемый девятый бит и стоп-бит. При передаче девятый бит может использоваться для повышения достоверности передачи путем контроля по четности и в него можно поместить значение признака паритета из PSW. Частота приема/передачи выбирается программно и может быть равна 1/32 или 1/64 частоты резонатора в зависимости от SMOD.

Режим 3. Совпадает с режимом 2, но частота приема/передачи является величиной переменной и задается таймером.

1.11.3 Регистр SCON

Регистр предназначен для управления режимом работы UART. Регистр содержит управляющие биты и девятый бит принимаемых или передаваемых данных RB8 и TB8, а также биты прерывания приемопередатчика RI и TI. Функциональное назначение битов указано в табл. 8 и 9.

8 Регистр управления/статуса UART

Символ	Разряд	Имя и назначение
SM0	SCON.7	Биты управления режимом работы UART. Устанавливаются/сбрасываются программно (табл. 9)
SM1	SCON.6	
SM2	SCON.5	Бит управления режимом UART. Устанавливается программно для запрета приема сообщения, в котором девятый бит равен 0
REN	SCON.4	Бит разрешения приема. Устанавливается/сбрасывается программно для разрешения/запрета приема последовательных данных
TB8	SCON.3	Передача бита 8. Устанавливается/сбрасывается программно для задания девятого передаваемого бита в режиме UART – 9 бит
RB8	SCON.2	Прием бита 8. Устанавливается/сбрасывается аппаратно для фиксации девятого принимаемого бита в режиме UART – 9 бит
TI	SCON.1	Флаг прерывания передатчика. Устанавливается аппаратно при окончании передачи байта. Сбрасывается программно после обслуживания прерывания
RI	SCON.0	Флаг прерывания приемника. Устанавливается аппаратно при приеме байта. Сбрасывается программно после обслуживания прерывания

9 Режим работы UART

SM0	SM1	Режим работы UART
0	0	Синхронный приемопередатчик 8 бит
0	1	UART – 8 бит. Изменяемая скорость передачи
1	0	UART – 9 бит. Фиксированная скорость передачи
1	1	UART – 9 бит. Изменяемая скорость передачи

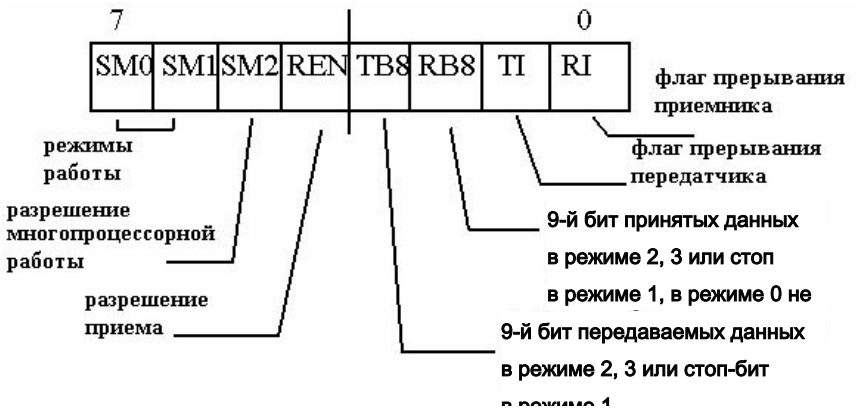
Прикладная программа путем загрузки в два старших разряда SCON определяет режим работы UART. Во всех режимах передача инициируется любой командой, где SBUF указан как получатель байта. Прием в UART в режиме 0 происходит при условии RI = 0 и REN = 1. В режимах 1 – 3 прием начинается с приходом старт-бита, если REN = 1.

В TB8 программно устанавливается значение девятого бита данных, который будет передан в режиме 2 или 3. В 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 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 – 3. Подпрограмма обслуживания этого 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 – 3. Подпрограмма обслуживания прерывания должна сбрасывать бит RI.

В виде байта регистр SCON можно изобразить следующим образом:



1.11.4 Скорость приема/передачи

Скорость зависит от режима работы последовательного порта и тактовой частоты микроконтроллера $f_{рез}$.

В режиме 0 частота зависит только от резонатора: $f_0 = f_{рез} / 12$. За один машинный цикл передается один бит.

В режимах 1 – 3 скорость зависит от значения управляющего бита SMOD в регистре специальных функций PCON (табл. 10).

В режиме 2 частота передачи $f_2 = (2^{SMOD}/64) f_{рез}$.

10 Регистр управления мощностью PCON

Символ	Разряд	Наименование и функция
SMOD	PCON.7	Удвоенная скорость передачи. Если бит установлен в 1, то скорость передачи вдвое больше, чем при SMOD = 0
–	PCON.6-4	Не используются
GF1	PCON.3	Флаги, специфицируемые пользователем (флаги общего назначения)
GF0	PCON.2	
PD	PCON.1	Бит пониженной мощности. При установке в 1 микроконтроллер переходит в режим пониженного энергопотребления
IDL	PCON.0	Бит холостого хода. Если бит установлен в 1, то микроконтроллер переходит в режим холостого хода

П р и м е ч а н и е. При одновременной записи 1 в PD и IDL бит PD имеет преимущество. Сброс PCON выполняется путем загрузки в него кода 0XXX0000.

В режимах 1 и 3 в формировании частоты приема/передачи, кроме управляющего бита SMOD, принимает участие таймер 1. При этом частота приема/передачи зависит от частоты переполнения (OVT1) и определяется следующим образом: $f_{1,3} = (2^{SMOD}/32) f_{OVT1}$. 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 приема/передачи определяется выражением $f_{1,3} = (2^{SMOD}/32)(f_{рез}/12)\ (256 - (TH1))$.

В табл. 11 приводится описание способов настройки Т/С1 для получения типовых частот передачи данных через UART.

11 Настройка таймера 1 для управления частотой работы UART

Частота приема/ передачи (BAUD RATE)		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

Продолжение табл. 11

Частота приема/ передачи (BAUD RATE)		Частота резонатора, МГц	SMOD	Таймер/счетчик 1		
				C/T	Режим (MODE)	Перезагружаемое число
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 Гц	11,059	0	0	2	1DH
	110 Гц	6	0	0	2	72H
	110 Гц	12	0	0	1	0FEEBH

1.12 СИСТЕМА ПРЕРЫВАНИЙ

Система прерываний (рис. 7) включает 5 источников прерываний с фиксированными векторами, из которых 2 внешних (входы INT0, INT1) и 3 внутренних (таймеры 0 и 1, последовательный порт) источника.

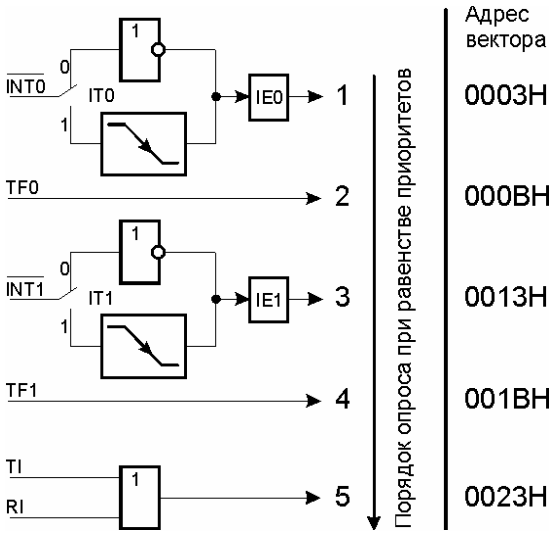


Рис. 7 Схема прерываний

Внешние прерывания INT0 и INT1 могут быть вызваны уровнем или переходом сигнала из 1 в 0 (срезом) на входах микроконтроллера в зависимости от значений управляющих битов IT0 и IT1 в регистре TCON. От внешних прерываний устанавливаются флаги IE0 и IE1 в регистре TCON, которые инициируют вызов соответствующей подпрограммы обслуживания прерывания. Сброс этих флагов выполняется аппаратно только в том случае, если прерывание было вызвано по переходу (срезу) сигнала. Если же прерывание вызвано уровнем входного сигнала, то сбросом флага IE управляет соответствующая подпрограмма обслуживания прерывания путем воздействия на источник прерывания с целью снятия им запроса.

Флаги запросов прерываний от таймеров TF0 и TF1 сбрасываются автоматически при передаче управления подпрограмме обслуживания.

Флаги запросов прерываний RI и TI устанавливаются аппаратно, но сбрасываться должны программой. Прерывания от последовательного порта при передаче и приеме вызывают одну и туже подпрограмму обслуживания, в которой, опросив флаги, можно определить источник прерываний.

Прерывания могут быть вызваны или отменены программой, так как все перечисленные флаги программно доступны.

В блоке регистров специальных функций есть два регистра, предназначенных для управления режимом прерываний и уровнями приоритета. Форматы этих регистров, имеющих символические имена IE и IP, описаны в табл. 12 и 13 соответственно.

12 Регистр масок прерываний IE

Символ	Разряд	Имя и назначение
EA	IE.7	Общее разрешение прерываний. Сбрасывается программно для запрета всех прерываний независимо от состояний IE4 – IE0
–	IE.6, 5	Не используются
ES	IE.4	Бит разрешения прерываний от UART. Установка/сброс программой для разрешения/запрета прерываний от флагов TI, RI
ET1	IE.3	Бит разрешения прерываний от таймера 1. Установка/сброс программой для разрешения/запрета прерываний от таймера 1
EX1	IE.2	Бит разрешения внешнего прерывания 1. Установка/сброс программой для разрешения/запрета прерываний

ET0	IE.1	Разрешение прерываний от таймера 0. Работает аналогично IE.3
EX0	IE.0	Разрешения внешнего прерывания 0. Работает аналогично IE.2

13 Регистр приоритетов прерываний IP

Символ	Разряд	Имя и назначение
–	IP.7 – 5	Не используются
PS	IP.4	Бит приоритета UART. Установка/сброс программой для назначения прерыванию от UART высшего/низшего приоритета
PT1	IP.3	Бит приоритета таймера 1. Установка/сброс программой для назначения прерыванию от таймера 1 высшего/низшего приоритета
PX1	IP.2	Бит приоритета внешнего прерывания 1. Установка/сброс программой для назначения прерыванию INT1 высшего/низшего приоритета
PT0	IP.1	Бит приоритета таймера 0. Работает аналогично IP.3
PX0	IP.0	Приоритет внешнего прерывания 0. Работает аналогично IP.2

Возможность программной установки/сброса любого управляющего бита в этих двух регистрах делает систему прерываний исключительно гибкой.

Приоритет источника прерывания определяется порядком опроса флагов прерываний и производится в два этапа. На первом – опрашиваются те, источники которых получили высший приоритет с помощью регистра IP, а на последнем – низший. Для каждого из этапов очередность опроса приведена на рис. 7.

Флаги прерываний опрашиваются в каждом машинном цикле. Ранжирование прерываний по приоритету выполняется в течение следующего машинного цикла. Прерывание сформируется, если оно не заблокировано одним из условий:

1) в данный момент обслуживается запрос прерывания равного или более высокого уровня приоритета;
2) текущий машинный цикл – не последний в цикле выполняемой команды;
3) выполняется команда RETI или любая команда, связанная с обращением к регистрам IE или IP.

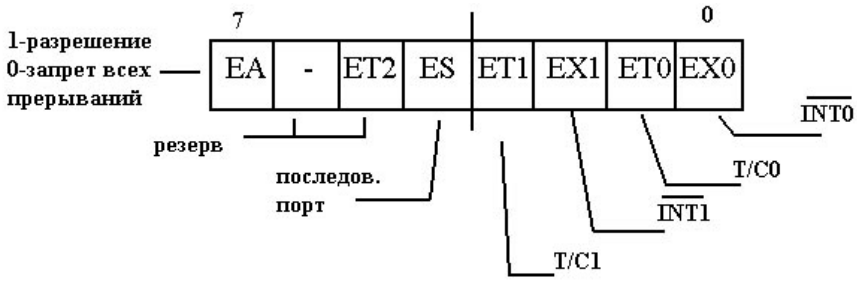
П р и м е ч а н и е. Если флаг прерывания был установлен, но по одному из перечисленных условий не получил обслуживания и к моменту окончания блокировки уже был сброшен, то запрос прерывания теряется.

По аппаратно сформированному коду команды LCALL (вызов подпрограммы) система прерываний помещает в стек содержимое программного счетчика PC и загружает в PC адрес вектора прерывания соответствующей подпрограммы обслуживания. По этому адресу должна быть расположена команда безусловного перехода JMP к начальному адресу подпрограммы обслуживания прерывания. Эта подпрограмма в случае необходимости должна начинаться командами записи в стек PUSH слова состояния программы PSW, аккумулятора A, расширителя аккумулятора B, указателя данных DPTR и т.д. и заканчиваться командами восстановления из стека POP. Подпрограммы обслуживания прерывания обязательно завершаются командой RETI, по которой в программный счетчик перезагружается из стека сохраненный адрес возврата в основную программу. Команда RET также возвращает управление, но при этом не снимает блокировку прерывания.

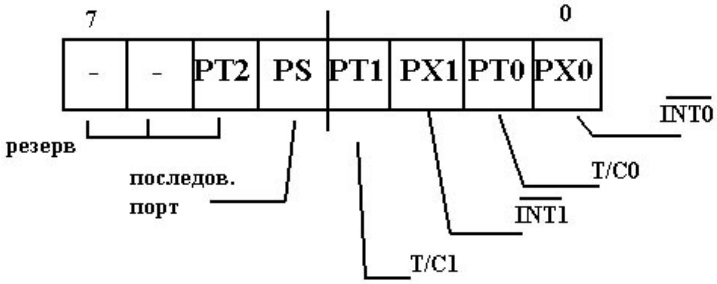
Ввиду того, что адреса векторов прерываний расположены в памяти с интервалом 8 ячеек, по ним, как правило, размещают не сами процедуры обслуживания, а команды безусловного перехода на подпрограммы.

В виде байтов регистры IE и IP можно представить следующим образом:

Регистр IE:



Регистр IP:



2 СИСТЕМА КОМАНД МИКРОКОНТРОЛЛЕРА INTEL 8051

2.1 ОБЩИЕ СВЕДЕНИЯ

Система команд – это уникальный, характерный для данного микропроцессора набор команд (инструкций), определяющих перечень всех его возможных операций. Каждая инструкция для микропроцессора представляется в двоичном коде, который называется кодом операции (КОП).

В зависимости от числа использованных кодов операций системы команд микропроцессоров подразделяют на два вида: CISC и RISC. Термин CISC является аббревиатурой английского определения Complex Instruction Set Computer и означает сложную (полную) систему команд. Аналогично термин RISC означает сокращенную систему команд и происходит от английского Reduced Instruction Set Computer.

Систему команд микроконтроллера INTEL 8051 можно отнести к типу CISC. Система содержит 111 базовых команд (при общем количестве 255), которые по функциональному признаку могут быть разделены на пять групп:

- команды передачи данных,
- арифметические операции,
- логические операции,
- операции с битами,
- команды передачи управления.

94 команды, т.е. большинство, имеют формат в один или два байта и выполняются за один или два машинных цикла. При тактовой частоте 12 МГц длительность машинного цикла составляет 1 мкс.

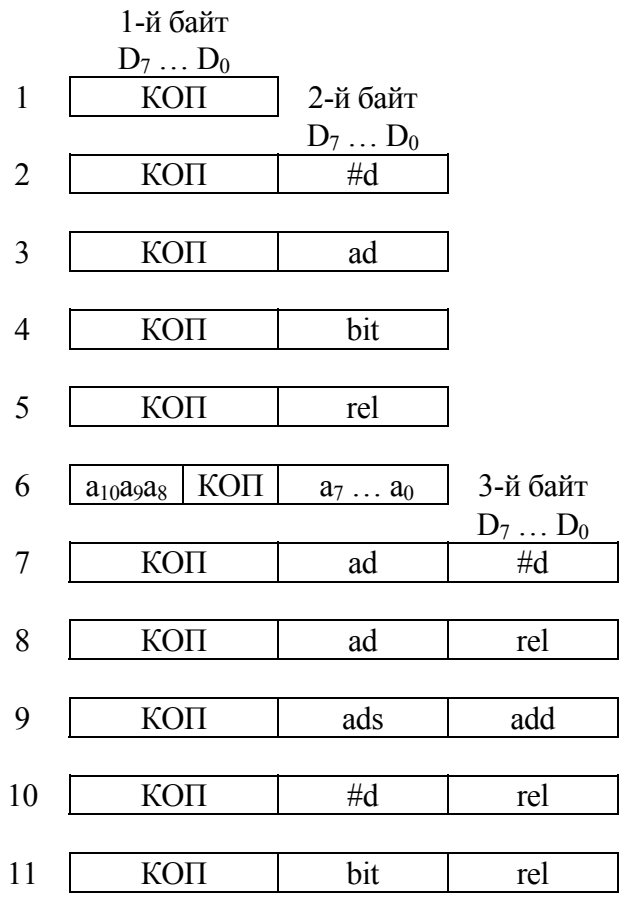
В приложении приведены ассемблерная мнемоника, описание команд и их характеристики: тип (Т), число байтов в командах (Б), а также продолжительность исполнения команд в циклах (Ц).

2.1.1 Типы команд

На рис. 8 показаны 13 типов команд. Первый байт команды любых типа и формата всегда содержит код операции (КОП). Второй и третий байты содержат либо адреса операндов, либо непосредственные операнды.

Приняты следующие обозначения:

A – аккумулятор	(x) – содержимое элемента x
R_i – регистр выбранного банка	((x)) – содержимое по адресу, хранящемуся в x
i – номер регистра	(x)[M] – разряд M элемента x
direct – прямо адресуемый 8-битовый внутренний адрес	
@ R_i – косвенно адресуемая 8-битовая ячейка ОЗУ	/x – инверсия
d8, #d – 8-битовое непосредственное данное	ads – адрес источника данных
d16 – 16-битовое непосредственное данное	add – адрес приемника данных
dH, dL – старшие, младшие биты непосредственных 16-битных данных	ad16h – старший байт адреса
adr11 – 11-битовый адрес	ad16l – младший байт адреса
adr16 – 16-битовый адрес	#d16h – старший байт данных
adrL – младшие биты адреса	ad – 8-разрядный адрес
disp8, rel – 8-битовый байт смещения	#d16l – младший байт данных
bit – прямо адресуемый бит	



12	КОП	ad16h	ad16l
13	КОП	#d16h	#d16l

Рис. 8 Типы команд

2.1.2 Типы операндов

Состав операндов включает в себя операнды четырех типов: биты, 4-битные цифры (тетрады), байты и 16-битные слова.

Адреса	(D_7)							(D_0)
7FH								
2FH	7F	7E	7D	7C	7B	7A	79	78
2EH	77	76	75	74	73	72	71	70
2DH	6F	6E	6D	6C	6B	6A	69	68
2CH	67	66	65	64	63	62	61	60
2BH	5F	5E	5D	5C	5B	5A	59	58
2AH	57	56	55	54	53	52	51	50
29H	4F	4E	4D	4C	4B	4A	49	48
28H	47	46	45	44	43	42	41	40
27H	3F	3E	3D	3C	3B	3A	39	38
26H	37	36	35	34	33	32	31	30
25H	2F	2E	2D	2C	2B	2A	29	28
24H	27	26	25	24	23	22	21	20
23H	1F	1E	1D	1C	1B	1A	19	18
22H	17	16	15	14	13	12	11	10
21H	0F	0E	0D	0C	0B	0A	09	08
20H	07	06	05	04	03	02	01	00
1FH	Банк 3							
18H								
17H	Банк 2							
10H								
0FH	Банк 1							
08H								
07H	Банк 0							
00H								

Рис. 9 Карта адресуемых битов в резидентной памяти данных

Микроконтроллер в RDM имеет 128 программно-управляемых флагов пользователя – битов. Имеется также возможность адресации отдельных битов блока регистров специальных функций, включая порты ввода/вывода. Для адресации битов используется прямой 8-битный адрес (bit). Косвенная адресация битов невозможна. Карты адресов отдельных битов представлены на рис. 9 и 10.

Четырехбитные операнды используются только при операциях обмена SWAP и XCHD.

Прямой адрес бита (D_7)								(D_0)	Имя регистра
0FFH									
0F0H	F7	F6	F5	F4	F3	F2	F1	F0	B
0E0H	E7	E6	E5	E4	E3	E2	E1	E0	A
0D0H	D7	D6	D5	D4	D3	D2	D1	D0	PSW
0B8H	–	–	–	BC	BB	BA	B9	B8	IP
0B0H	B7	B6	B5	B4	B3	B2	B1	B0	P3
0A8H	AF	–	–	AC	AB	AA	A9	A8	IE
0A0H	A7	A6	A5	A4	A3	A2	A1	A0	P2
98H	9F	9E	9D	9C	9B	9A	99	98	SCON
90H	97	96	95	94	93	92	91	90	P1
88H	8F	8E	8D	8C	8B	8A	89	88	TCON
80H	87	86	85	84	83	82	81	80	P0

Рис. 10 Карта адресуемых битов в блоке регистров специальных функций

Восьмибитным операндом (байтом) может быть ячейка памяти программ (ПП), памяти данных (резидентной или внешней), константа (непосредственный операнд), регистры специальных функций, включая порты ввода/вывода. Порты и регистры специальных функций адресуются только прямым способом. Байты памяти могут адресоваться также и косвенным образом через адресные регистры R0, R1, DPTR и PC.

Двухбайтные операнды – это константы и прямые адреса, для представления которых используются второй и третий байты команды.

2.1.3 Способы адресации данных

В микроконтроллере используются следующие способы адресации данных:

- неявный;
- регистровый;
- непосредственный;
- прямой;
- косвенный.

Неявный способ получил такое название из-за того, что адрес операнда в команде явно не указывается, а подразумевается самим кодом операции (КОП). Например, однобайтная команда CLR C (сброс флага переноса) представляет собой только КОП. Этот способ адресации позволяет получать команды минимального формата.

Регистровый способ адресации используется для операндов, хранящихся в одном из регистровых банков: регистры общего назначения R0 – R7. Например, команда DEC R0 (декремент содержимого регистра R0). Этот способ адресации также позволяет получать команды минимального формата.

Непосредственный способ адресации служит для использования в качестве операнда непосредственных данных. При этом операнд находится в программной памяти непосредственно за КОП команды. Данные могут быть одно- или двухбайтовыми. Например, команда MOV A, #d (загрузить байт в аккумулятор) имеет двухбайтный формат: первый байт КОП и байт данных #d. Например, команда MOV DPTR, #d16 (загрузить два байта в регистр-указатель DPTR) имеет трехбайтный формат: первый байт КОП и два байта данных #d16.

Прямой способ адресации предполагает указание операндов посредством адреса, содержащегося в команде. Адрес может быть одно- или двухбайтовым. Например, команда ADD A, ad (сложение аккумулятора с содержимым памяти по адресу ad) имеет двухбайтный формат: первый байт КОП и байт адреса ad. Например, команда LJMP ad16 (длинный переход по адресу ad16) имеет трехбайтный формат: первый байт КОП и два байта адреса ad16.

Косвенный способ адресации предполагает указание операндов посредством адреса, содержащегося в регистре либо в регистровой паре. В команде указывается регистр, который в свою очередь указывает адрес операнда. Например, команда MOV A, @R0 (загрузить в аккумулятор содержимое ячейки внутренней памяти, восьмиразрядный адрес которой содержится в регистре R0). Например, команда MOV A, @ DPTR (загрузить в аккумулятор содержимое ячейки внешней памяти, адрес которой содержится в двухбайтном регистре-указателе DPTR). Этот способ адресации позволяет уменьшить формат команд и повысить гибкость программирования.

Многие команды для указания операндов комбинируют различные способы адресации. Например, команда MOV @R0, #d (загрузить байт данных #d в ячейку внутренней памяти, восьмиразрядный адрес которой содержится в регистре R0). Здесь используются непосредственный (источник операнда) и косвенный (приемник операнда) способы адресации.

2.1.4 Флаги результата

Слово состояния программы (регистр PSW) включает в себя четыре флага (см. табл. 2):

CY – перенос,

AC – вспомогательный перенос (полуперенос),

OV – переполнение,

P – паритет.

Флаг CY устанавливается, если в старшем бите результата возникает перенос или заем. При выполнении операций умножения и деления флаг CY сбрасывается.

Флаг AC устанавливается, если при выполнении операции сложения или вычитания между тетрадами байта (полубайтами) возник перенос или заем.

Флаг OV устанавливается, если результат операции сложения или вычитания не укладывается в семи битах и старший (восьмой) бит результата не может интерпретироваться как знаковый. При выполнении операции деления флаг OV сбрасывается, а в случае деления на нуль устанавливается. При умножении флаг OV устанавливается, если результат больше 255.

Флаг паритета напрямую зависит от текущего значения аккумулятора. Если число единичных битов аккумулятора нечетное, то флаг P устанавливается, а если четное – сбрасывается. При нулевом значении аккумулятора P = 0. Все попытки изменить флаг P, присваивая ему новое значение, бесполезны, если содержимое аккумулятора при этом останется неизменным.

14 Команды, модифицирующие флаги результата

Команды	Флаги	Команды	Флаги
ADD	CY, OV, AC	CLR CY	CY = 0
ADDC	CY, OV, AC	CPL CY	CY = NOT(CY)
SUBB	CY, OV, AC	ANL CY, b	CY

MUL	CY = 0, OV	ANL CY, /b	CY
DIV	CY = 0, OV	ORL CY, b	CY
DA	CY	ORL CY, /b	CY
RRC	CY	MOV CY, b	CY
RLC	CY	CJNE	CY
SETB C	CY = 1		

В табл. 14 перечисляются команды, при выполнении которых модифицируются флаги результата. В таблице отсутствует флаг паритета, так как его значение изменяется всеми командами, которые изменяют содержимое аккумулятора. Кроме команд, приведенных в таблице, флаги модифицируются командами, в которых местом назначения результата определены PSW или его отдельные биты, а также командами операций над битами.

2.1.5 Символическая адресация

При использовании ассемблера для получения объектных кодов программ допускается применение в программах символических имен регистров специальных функций, портов и их отдельных битов (рис. 10).

Для адресации отдельных битов и портов (такая возможность имеется не у всех регистров специальных функций) можно использовать символическое имя бита следующей структуры: <имя регистра или порта>.<номер бита>.

Например, символическое имя пятого бита аккумулятора будет следующим: ACC.5. Символические имена являются зарезервированными словами, и их не надо определять с помощью директив ассемблера.

2.2 КОМАНДЫ ПЕРЕДАЧИ ДАННЫХ

Большую часть команд данной группы (табл. П.1) составляют команды передачи и обмена байтов. Команды пересылки битов представлены в группе команд битовых операций. Все команды данной группы не модифицируют флаги результата, за исключением команд загрузки PSW и аккумулятора (флаг паритета).

2.2.1 Структура информационных связей

В зависимости от способа адресации и места расположения операнда можно выделить девять типов операндов, между которыми возможен информационный обмен. Граф возможных операций передачи данных показан на рис. 11. Передачи данных могут выполняться без участия аккумулятора.

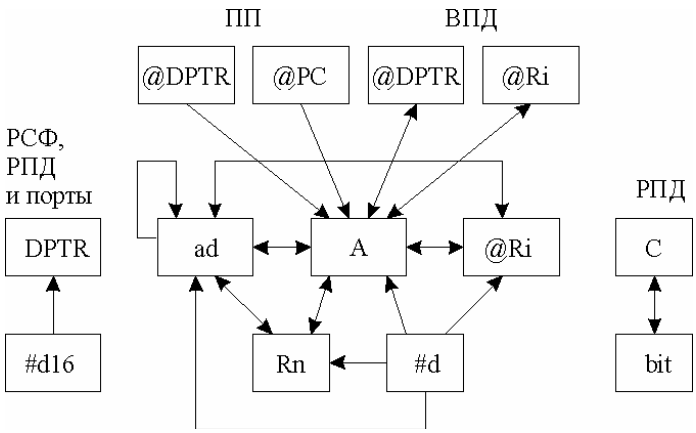


Рис. 11 Граф путей передачи данных

2.2.2 Обращение к аккумулятору

Обращение к аккумулятору может быть выполнено с использованием неявной и прямой адресации. В зависимости от способа адресации аккумулятора применяется одно из символических имен: A или ACC (прямой адрес). При прямой адресации обращение к аккумулятору производится как к одному из регистров специальных функций, и его адрес указывается во втором байте команды. Использование неявной адресации аккумулятора предпочтительнее, но не всегда возможно, например, при обращении к отдельным битам аккумулятора.

2.2.3 Обращение к внешней памяти данных

MOVX <байт_приемника>, <байт_источника>

При использовании команд MOVX @Ri обеспечивается доступ к 256 байтам внешней памяти данных. Существует также режим обращения к расширенной внешней памяти данных, когда для доступа используется 16-битный адрес, хранящийся в регистре-указателе данных DPTR. Команды MOVX @DPTR обеспечивают доступ к 65536 байтам внешней памяти данных.

1) (A): = ((R_i));
   i = 0, 1.

   П р и м е р: A = 32h, R0 = 83h, Внешнее ЗУ[83h] = B6h
   MOVX A, @R0 → A = B6h.

2) (A): = ((DPTR)).

   П р и м е р: A = 5Ch, DPTR = 1ABEh, Внешнее ЗУ[1ABEh] = 72h
   MOVX A, @DPTR→ A = 72h.

3) ((R_i)): = (A);
   i = 0, 1.

   П р и м е р: A = 95h, R1 = FDh, Внешнее ЗУ[FDh] = 00h
   MOVX @R1, A→ Внешнее ЗУ[FDh] = 95h.

4) ((DPTR)): = (A).

   П р и м е р: A = 97h, DPTR = 1FFFh, Внешнее ЗУ[1FFFh] = 00h
   MOVX @DPTR, A→ Внешнее зу[1FFFh] = 97h.

2.2.4 Обращение к памяти программ

MOVC A, @A+(<R16>)

Команда MOVC выполняет считывание в аккумулятор содержимого программной памяти. Это позволяет использовать ПЗУ программ для хранения констант. Способ адресации ячейки программной памяти косвенный. В качестве указателя адреса используется программный счетчик PC и регистр-указатель данных DPTR.

1) (A): = ((A)+(DPTR)).

   П р и м е р: A = 1Bh, DPTR = 1020h, ПЗУ[103B] = 48h
   MOVC A, @A + DPTR → A = 48h.

2) (A): = ((A)+(PC)).

П р и м е р: A = FAh, PC = 0289h, ПЗУ[0384] = 9Bh
MOVC A, @A + PC → A = 9Bh, PC = 028Ah.

2.2.5 Обращение к стеку

Для работы со стеком служат две команды PUSH и POP. Первая команда помещает прямоадресуемый байт в стек, вторая – наоборот извлекает данные из стека. Адресация стека осуществляется косвенно через восьмиразрядный регистр указатель стека SP, который автоматически модифицируется после каждого обращения к стеку. Команда PUSH инкрементирует SP, а POP декрементирует его.

POP <direct>

(direct): = ((SP)), (SP): = (SP)-1.

П р и м е р: SP = 32h, DPH = 01, DPL = ABh,
ОЗУ[32] = 12, ОЗУ[31] = 56h, ОЗУ[30] = 20h
POP DPH
POP DPL → SP = 30h, DPH = 12, DPL = 56
POP SP → SP = 20.

PUSH <direct>

(SP): = (SP)+1, ((SP)): = <direct>

П р и м е р: SP = 09h, DPTR = 1279h
PUSH DPL
PUSH DPH → SP = 0Bh, ОЗУ[0A] = 79h, ОЗУ[0B] = 12h

2.3 АРИФМЕТИЧЕСКИЕ ОПЕРАЦИИ

Данную группу образуют 24 команды (табл. П.2), выполняющие операции сложения, десятичной коррекции, инкремента/декремента байтов. Имеются команды вычитания, умножения и деления байтов.

Команды ADD и ADDC допускают сложение аккумулятора с большим числом операндов. Аналогично командам ADDC существуют четыре команды SUBB, что позволяет достаточно просто производить вычитание байтов и многобайтных двоичных чисел.

В микроконтроллере реализуется расширенный список команд инкремента/декремента байтов, команда инкремента 16-битного регистра-указателя данных.

ADD A, <байт-источник>

1) (A): = (A)+(Ri).

П р и м е р: A = C3h, R6 = AAh
ADD A, R6 → A = 6Dh, R6 = AAh, CY = 1, AC = 0, OV = 1.

2) (A): = (A)+((Ri)).

П р и м е р: A = 95h, R1 = 35h ОЗУ[35] = 4Ch
ADD A, @R1 → A = E1h, CY = 0, AC = 1, OV = 0.

3) (A): = (A)+(direct).

П р и м е р: A = 77h, ОЗУ[90] = FFh
ADD A, 90h → A = 76h, CY = 1, AC = 1, OV = 0.

4) (A): = (A)+#data.

П р и м е р: A = 09h
ADD A, #0D3h →A = DCh, CY = 0, AC = 0, OV = 0.

ADDC A, <байт-источник>

1) (A): = (A)+(C)+(Ri).

П р и м е р: A = B2h, R3 = 99, CY = 1
ADDC A, R3 → A = 4Ch, CY = 1, AC = 0, OV = 1.

2) (A): = (A)+(C)+((Ri)).

П р и м е р: A = D5h, R0 = 3Ah, ОЗУ[3A] = 1Ah, CY = 1
ADDC A, @R0 → A = F0h, CY = 0, AC = 1, OV = 0.

3) (A): = (A)+(C)+(direct).

П р и м е р: A = 11h, ОЗУ[80] = DFh, CY = 1
ADDC A, 80h →A = F1h, CY = 0, AC = 1, OV = 0.

4) (A): = (A)+(C)+#data.

П р и м е р: A = 55h, CY = 0
ADDC A, #55h A = AAh, CY = 0, AC = 0, OV = 1.

DA A
десятичная коррекция bin->bcd

П р и м е р: 1) A = 56h, R3 = 67h, CY = 1
ADDC A, R3 → A = BEh, CY = 0
DA A → A = 24h, CY = 1.
2) A = 30h, CY = 0
ADD A, #99h → A = C9h, CY = 0
DA A → A = 29h, CY = 1.

DEC <байт – 1

1) DEC A.

П р и м е р: A = 11h, CY = 1, AC = 1
DEC A → A = 10h, CY = 1, AC = 1.

2) DEC (Rn);
n = 0, ..., 7;
(Rn) = (Rn)-1.

П р и м е р: R1 = 7Fh, ОЗУ[7F] = 40h, ОЗУ[7E] = 00h;
DEC @R1
DEC R1
DEC @R1 → R1 = 7Eh, ОЗУ[7F] = 3Fh, ОЗУ[7E] = FFh.

3) DEC <direct>;
(direct): = (direct)-1.

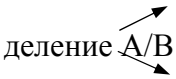
П р и м е р: SCON = A0h, C = 1, AC = 1

DEC SCON SCON = 9Fh, CY = 1, AC = 1.

4) DEC @Ri;
((Ri)) = ((Ri)-1).

П р и м е р: R1 = 7Fh, ОЗУ[7F] = 40h, ОЗУ[7E] = 00h;
DEC @R_1
DEC R_1
DEC @R_1 → R_1 = 7Eh, ОЗУ[7F] = 3Fh, ОЗУ[7E] = FFh.

DIV AB



A: целая часть

B: остаток

(A): = ((A)/(B)) [15-8];
(B): = ((A)/(B)) [7-0].

П р и м е р: A = 251 = FBh = 11111011b;
B = 18 = 12h = 00010010b;

DIV AB
A = 13 = 0Dh = 00001101b;
B = 17 = 11h = 00010001b; т. к. 251 = (13*18)+17
CY = 0, OV = 0.

INC <байт> +1

1) INC A;
(A): = (A)+1.

П р и м е р: A = 1Fh, AC = 0
INC A → A = 20h, AC = 0.

2) INC Rn;
n = 0, ..., 7.

П р и м е р: R4 = FFh, CY = 0, AC = 0
INC R4 → R4 = 00h, CY = 0, AC = 0.

3) INC <direct>;
<direct> : = <direct> + 1.

П р и м е р: ОЗУ[43] = 22h
INC 43h → ОЗУ[43] = 23h.

4) INC @Ri;
i = 0, 1.

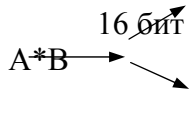
П р и м е р: R1 = 41h, ОЗУ[41] = 4Fh, AC = 0
INC @R1 →R1 = 41h, ОЗУ[41] = 50h, AC = 0.

INC DPTR

(DPTR): = (DPTR)+1.

П р и м е р: DPH = 12h, DPL = FEh
INC DPTR
INC DPTR
INC DPTR → DPH = 13h DPL = 01h.

MUL A B



П р и м е р: A = 80 = 50h, B = 160 = A0h, CY = 1, OV = 0
MUL A B
A = 00h, B = 50 = 32h, CY = 0, OV = 1.

SUBB A, <байт источника>
(A):=(A) - (C) – <байт источника>

1) (A): = (A) - (C) - (Ri);
i = 0, ..., 7.

П р и м е р: A = C9h, R2 = 54h, CY = 1
SUBB A, R2 → A = 74h, R2 = 54h, CY = 0,
AC = 0, OV = 1.

2) (A): = (A) - (C) - (direct).

П р и м е р: A = 97h, R2 = 25h, C = 0
SUBB A, B → A = 72h, CY = 0,
AC = 0, OV = 1.

3) (A): = (A) - (C) - ((Ri));
i = 0, 1.

П р и м е р: A = 49h, C = 1, R0 = 33h, ОЗУ[33] = 68h
SUBB A, @R0 → A = E0h, CY = 1,
AC = 0, OV = 0.

4) (A): = (A) - (C) - (#data8).

П р и м е р: A = 0BEh, CY = 0
SUBB A, #3F → A = 7Fh, CY = 0,
AC = 1, OV = 1.

2.4 ЛОГИЧЕСКИЕ ОПЕРАЦИИ

Данную группу образуют 25 команд (табл. П.3), реализующих функционально полную систему логических операции над байтами. В микроконтроллере расширено число типов операндов, участвующих в операциях.

Имеется возможность производить операцию "исключающее ИЛИ" с содержимым портов. Команда XRL может быть эффективно использована для инверсии отдельных битов портов.

ANL <байт_назначения>, <байт_источник>
(логическое И)

1) (A): = (A)and(Ri); ANL A, Rn;
i = 0, ..., 7.

П р и м е р: A = FEh, R2 = C5h
ANL A, R2 →A = C4h.

2) (A): = (A)and(direct); ANL A, < direct >.

П р и м е р: A = 0A3h, PSW = 86h
ANL A, PSW →A = 82h.

3) (A): = (A)and((Ri)); ANL A, @Rn;
i = 0, 1.

П р и м е р: A = 0BCh, ОЗУ[35] = 47h, R0 = 35h
ANL A,@R0 →A = 04h.

4) (A): = (A)and #data; ANL A, #data.

П р и м е р: A = 36h
ANL A, #0DDh →A = 14h.

5) (direct): = (direct)and(A); ANL < direct >, A.

П р и м е р: A = 55h, P2 = 0AAh
ANL P2, A →P2 = 00h.

6) (direct): = (direct)and #data; ANL < direct >, #data.

П р и м е р: P1 = FFh
ANL P1, #73h →P1 = 73h.

ANL C, <бит_источника>
(только прямая адресация**)**

1) (C): = (C)and(bit).

П р и м е р: CY = 1, P1.0 = 0
ANL C, P1.0 →CY = 0.

2) (C): = (C)and(/bit).

П р и м е р: C = 1, AC = 0
ANL C, /AC →CY = 1, AC = 0.

CLR A (сброс аккумулятора в 0)

1) (A): = 0.

П р и м е р: (A) = 6DH, CY = 0, AC = 1
CLR A → (A) = 0, CY = 0, AC = 1.

CLR<bit> (сброс бита)

1) (C): = 0; CY = 1, CLR C→ CY = 0.
2) (bit): = 0.

П р и м е р: P1 = 5Eh(01011110b)
CLR P1.3→ P1 = 56h(01010110b).

CPL Аинверсия аккумулятора

1) (A): = /(A).

П р и м е р: (A) = 65H = 01100101b
CPL A → (A) = 9AH = 10011010.

CPL<bit>

1) (bit): = /(bit). P1 = 39h (00111001b)
CPL P1.1
CPL P1.3→ P1 = 33h (00110011).

2) (C): = /(C). CY = 0, AC = 1, OV = 0
CPL C→ CY = 1, AC = 1, OV = 0.

ORL <байт_назначения**>**, <байт_источника>
Логическое ИЛИ

1) (A): = (A) OR (Ri);
i = 0, …, 7.

П р и м е р: A = 15h, R5 = 6Ch
ORL A, R5 → A = 7Dh.

2) (A): = (A) OR (direct).

П р и м е р: A = 84h, PSW = 0C2h
ORL A, PSW→ A = C6h.

3) (A): = (A) OR ((Ri));
i = 0, 1.

П р и м е р: A = 52h, R0 = 6Dh, ОЗУ[6D] = 49h
ORL A,@R0→ A = 5Bh.

4) (A) = (A) OR # <data>.

П р и м е р: A = F0h
ORL A, #0Ah→ A = Fah.

5) (direct): = (direct) OR (A).

П р и м е р: A = 34h, IP = 23h

ORL IP, A→ IP = 37h.

6) (direct): = (direct) OR # <data>.

П р и м е р: P1 = 00h
ORL P1, #0C4h→ P1 = C4h.

RL A
циклический сдвиг аккумулятора влево

A = 0D5h; CY = 0.

П р и м е р: RL A→ A = 0ABh, CY = 0.

RLC A
циклический сдвиг влево через бит C

RR A
циклический сдвиг вправо

RRC A
циклический сдвиг вправо через бит C
XRL <байт_назначения>,<байт_источника>
Исключающее ИЛИ

1) (A): = (A) XOR (Ri);
i = 0, ..., 7.

П р и м е р: A = C3h, R6 = 0AAh
XRL A, R6→ A = 69h.

2) (A): = (a) XOR (direct) .

П р и м е р: A = 0Fh, P1 = 0A6h
XRL A,P1→ A = A9h.

3) (A): = (A) XOR ((Ri));
i = 0,1.

П р и м е р: A = 55h, R1 = 77h, ОЗУ[77] = 5Ah
XRL A,@R1→ A = 0Fh.

4) (A) = (A) XOR # <data>.

П р и м е р: A = 0C3h,
XRL A, 0F5h→ A = 36h.

5) (direct): = (direct) XOR (A).

П р и м е р: A = 31h, P1 = 82h
XRL P1, A → P1 = B3h.

6) (direct): = (direct) XOR # <data>.

П р и м е р: IP = 65h

XRL IP, #65h→ IP = 00h.

SWAP A
обмен тетрадами внутри А

П р и м е р: A = 0D7h → SWAP A → A = 7Dh.

2.5 КОМАНДЫ ПЕРЕДАЧИ УПРАВЛЕНИЯ

К данной группе команд (табл. П.4) относятся команды условного и безусловного ветвления, вызова подпрограмм и возврата из них, а также команда пустой операции NOP. В большинстве команд используется прямая адресация, т.е. адрес перехода целиком (или его часть) содержится в самой команде передачи управления. Можно выделить три разновидности команд ветвления по разрядности указываемого адреса перехода.

2.5.1 Длинный переход

Длинный переход – переход по всему адресному пространству памяти программ. В команде содержится полный 16-битный адрес перехода (ad16). Трехбайтные команды длинного перехода содержат в мнемокоде букву L (Long). Всего существуют две такие команды: LJMP – длинный переход и LCALL – длинный вызов подпрограммы. На практике редко возникает необходимость перехода в пределах всего адресного пространства, и чаще используются укороченные команды перехода, занимающие меньше места в памяти.

LCALL<addr 16>
длинный вызов

(PC): = (PC)+3;
(SP): = (SP)+1, ((SP)):=(PC[7÷0]);
(SP): = (SP)+1, ((SP)) :=(PC [15÷8]);
(PC): = < addr[15÷0]>.

П р и м е р: Пусть SP = 07h, адрес PRN = 1234h, адрес LCALL = 0126h.
После
LCALL PRN→ SP = 09h, PC 1234h,
(ОЗУ [08]) = 26h, (ОЗУ [09]) = 01h.

LJMP <addr 16>
длинный переход

(PC): = <addr [15÷0]>;
LJMP <метка>.

2.5.2 Абсолютный переход

Абсолютный переход – переход в пределах одной страницы памяти программ размером 2048 байтов. Такие команды содержат только 11 младших битов адреса перехода (ad11). Команды абсолютного перехода имеют формат 2 байта. Начальная буква мнемокода – А (Absolute). При выполнении команды в вычисленном адресе следующей по порядку команды ((PC) = (PC) + 2) 11 младших битов заменяются на ad11 из тела команды абсолютного перехода.

2.5.3 Относительный переход

Короткий относительный переход позволяет передать управление в пределах от – 128 до +127 байт относительно адреса следующей команды (команды, следующей по порядку за командой относительно-

го перехода). Существует одна команда короткого безусловного перехода SJMP (Short). Все команды условного перехода используют данный метод адресации. Относительный адрес перехода (rel) содержится во втором байте команды.

SJMP <метка>
короткий переход ±127 байт

(PC): = (PC)+2;
(PC): = (PC)+(rel 8).

2.5.4 Косвенный переход

Команда JMP @A + DPTR позволяет передавать управление по косвенному адресу. Эта команда удобна тем, что предоставляет возможность организации перехода по адресу, вычисляемому самой программой и неизвестному при написании исходного текста программы.

JMP @A + DPTR

(PC): = (A)[7÷0]+(DPTR[15÷0]).

П р и м е р: PC = 034Eh, A = 86h
DPTR = 0329h
JMP @A+ DPTR→ PC = 03Afh.

2.5.5 Условные переходы

Система условных переходов предоставляет возможность осуществлять ветвление по следующим условиям: аккумулятор содержит нуль (JZ), содержимое аккумулятора не равно нулю (JNZ), перенос равен единице (JC), перенос равен нулю (JNC), адресуемый бит равен единице (JB), адресуемый бит равен нулю (JNB).

Для организации программных циклов удобно пользоваться командой DJNZ. В качестве счетчика циклов может использоваться не только регистр, но и прямоадресуемый байт (например, ячейка резидентной памяти данных).

Команда CJNE эффективно используется в процедурах ожидания какого-либо события. Например, команда

WAIT: CJNE A, P0, WAIT

будет выполняться до тех пор, пока на линиях порта 0 не установится информация, совпадающая с содержимым аккумулятора.

Все команды данной группы, за исключением CJNE и JBC, не оказывают воздействия на флаги. Команда CJNE устанавливает флаг CY, если первый операнд оказывается меньше второго. Команда JBC сбрасывает флаг CY в случае перехода.

JNB<bit>, <rel8>
переход, если бит не установлен

JNB P1.3, LAB.

JNC<rel8>
переход, если бит C не установлен

JNC LAB.

JNZ<rel8>
переход, если A ≠ 0

JNZ LAB.

JZ<rel8>
переход, если A = 0

JZ LAB.

JB<bit>, <real>
переход, если установлен в 1

(PC): = (PC)+3;
if (bit) = 1 then PC = PC+3+<rel8>.

П р и м е р: JB ACC.2, LAB.

JBC<bit>, <real>
переход, если бит С установлен в 1 и сброс этого бита в 0

П р и м е р: JBC ACC.3, LAB3
JBC ACC.2, LAB2.

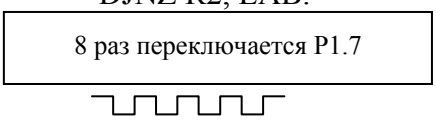
JC<rel8>
переход, если перенос установлен

П р и м е р: JC LAB.

DJNZ< байт**>,** <смещение>
декремент и переход, если не равно нулю

1) (PC): = (PC)+2; (Ri): = (Ri)-1; i = 0÷7;
if ((Ri)>0 OR (Ri)<0) then PC = PC+3+<rel8>.

П р и м е р: R2 = 08h, P1 = FFh
LAB: CPL P1.7
DJNZ R2, LAB.



2) (PC): = (PC)+3; (direct): = (direct)-1;
if ((direct)>0 OR (direct)<0) then PC = PC+3+<rel8>.

```
П р и м е р:    ОЗУ[40] = 01h, ОЗУ[50] = 80h, ОЗУ[60] = 25h
        DJNZ 40h, LAB1          ;переход на LAB2
        DJNZ 50h, LAB2
        DJNZ 60h, LAB3
              …
LAB1:   CLR  A
LAB2:   DEC  R1.
```

CJNE<байт_назначения>,<байт_источника>,<смещение>
сравнение и переход `если не равно`

1) (PC): = (PC)+3;
if (direct) < (A) then (PC): = (PC)+<rel8>, (C): = 0;
if (direct) > (A) then (PC):=(PC)+<rel8>, (C): = 1;
<rel 8> – число со знаком.

П р и м е р: A = 97h, P2 = F0h, CY = 0
CJNE A, P2, MT3
…
MT3: CLRA → A = 97h, P2 = F0h, CY = 1,
PC = PC+3+(rel8).

2) (PC): = (PC)+3;
if #data < (A) then (PC)+ <rel8>, (C): = 0;
if #data > (A) then (PC)+ <rel8>, (C): = 1.

П р и м е р: A = FCh, CY = 1
CJNE A, # 0BFh, MT4
…
MT4: JNC A → A = FDh, C=0, PC = PC+3+(rel8).

3) алгоритм тот же, но с Ri i = 0÷7.

П р и м е р: CJNE R7, #81h, MT5
…
MT5: NOP.

4) алгоритм тот же, но с @Ri, i = 0,1.

П р и м е р: CJNE @R0, #29h, MT6
…
MT6: DEC R0.

2.5.6 Подпрограммы

Для обращения к подпрограммам необходимо использовать команды вызова подпрограмм LCALL и ACALL. Эти команды в отличие от команд перехода LJMP и AJMP сохраняют в стеке адрес возврата в основную программу. Для возврата из подпрограммы необходимо выполнить команду RET. Команда RETI отличается от команды RET тем, что разрешает прерывания обслуженного уровня.

LCALL<adds 16>
длинный вызов

(PC): = (PC)+3;
(SP): = (SP)+1, ((SP)): = (PC[7÷0]);
(SP): = (SP)+1, ((SP)): = (PC [15÷8]);
(PC): = < addr[15÷0]>.

П р и м е р: Пусть SP = 07h, адрес PRN = 1234h, адрес LCALL = 0126h
После LCALL PRN→ SP=09h, PC 1234h
(ОЗУ [08])=26h, (ОЗУ [09]) = 01h.

ACALL <addr 11>
абсолютный вызов

адрес a0÷a10-11 бит;
(PC): = (PC)+2;
(SP): = (SP)+1, ((SP)): = (PC[7÷0]);
(SP): = (SP)+1, ((SP)): = (PC[15÷8]);
(PC[10÷0) = A10A9A8||A7A6…A0.

Пусть SP = 07h, метка MT1 по адресу 0345h, PC = 028Dh
028D: ACALL MT1 → SP = 09h, PC = 0345h
… ОЗУ[08] = 8Fh, ОЗУ[09] = 02h
0345: MT1…

RETI
возврат из прерывания

Восстанавливает счетчик команд PC, инициализирует логику прерываний.

RET

PC [15÷8] = ((SP));
(SP) = (SP)-1;
PC[7÷0] = ((SP))-1;
(SP) = (SP)-1.

2.6 ОПЕРАЦИИ С БИТАМИ

Отличительной особенностью данной группы команд (табл. П.5) является то, что они оперируют с однобитными операндами. В качестве таких операндов могут выступать отдельные биты некоторых регистров специальных функций и портов, а также 128 программных флагов пользователя.

Существуют команды сброса (CLR), установки (SETB) и инверсии (CPL) битов, а также конъюнкции и дизъюнкции бита и флага переноса. Для адресации битов используется прямой восьмиразрядный адрес (bit). Косвенная адресация битов невозможна.

ORL C, <бит_источника>

1) (C): = (C) OR (bit).

П р и м е р: CY = 0, P1 = 53h (01010011)
ORL C, P1.4→ CY = 1.

SETB <бит >
установить бит в 1

П р и м е р:
1) C = 0, SETB C→ CY = 1;
2) P2 = 38h.
SETB P2.0
SETB P2.7 → P2 = B9h.

MOV <бит_назначения>, <бит_источника>

1) (C):= (bit)

П р и м е р: C=0, P3=D5h (1101 0101 b)
MOV C, P3.0 → CY=1
MOV C, P3.3 → CY=0

MOV C, P3.7 → CY=1

2) (bit): = (C).

П р и м е р: CY = 1, P0 = 20h (0010 0000 b)
MOV P0.1, C
MOV P0.2, C
MOV P0.3, C → P0 = 2Eh (0010 1110 b).

Остальные команды были рассмотрены выше.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1 Перечислите характерные черты архитектуры однокристальных микроконтроллеров семейства MCS-51.

2 Укажите программно-доступные узлы микроконтроллера INTEL 8051 и назначение регистров специальных функций.

3 Назовите и охарактеризуйте четыре типа информационных объектов, с которыми может оперировать арифметико-логическое устройство микроконтроллера.

4 Какова организация памяти в микроконтроллере INTEL 8051?

5 Какие операции могут быть выполнены только с использованием аккумулятора?

6 Какие операции могут быть выполнены без участия аккумулятора?

7 Какой формат имеет слово состояния программы микроконтроллера INTEL 8051? Укажите назначение флагов.

8 Как переключить банк регистров общего назначения?

9 Какой регистр используется для адресации внешней памяти данных?

10 Какова длительность исполнения команд в микроконтроллере?

11 Охарактеризуйте режимы работы таймера/счетчика в микроконтроллере INTEL 8051.

12 Как с помощью таймера можно измерить длительность импульса?

13 Как с помощью таймера можно измерить время?

14 Каково назначение параллельных портов ввод/вывода?

15 Перечислите альтернативные функции портов.

16 Как выводится адрес при обращении к внешней памяти?

17 Охарактеризуйте режимы работы последовательного порта в микроконтроллере INTEL 8051.

18 Как изменить скорость передачи данных через последовательный порт?

19 Для чего используется девятый бит при передаче данных через последовательный порт?

20 Нарисуйте схему прерываний в микроконтроллере INTEL 8051. Перечислите и охарактеризуйте типы прерываний.

21 Для чего нужен регистр масок прерывания? Как изменить приоритеты прерываний?

22 Какие способы адресации реализует микроконтроллер INTEL 8051?

23 Охарактеризуйте способ адресации элементов стека в микроконтроллере.

24 Перечислите и охарактеризуйте группы команд микроконтроллера INTEL 8051.

25 Какой регистр выполняет функции базового регистра при косвенных переходах в программе?

СПИСОК ЛИТЕРАТУРЫ

1 Проектирование цифровых устройств на однокристальных микроконтроллерах / В.В. Сташин, А.В. Урусов, О.Ф. Мологонцева. М.: Энергоатомиздат, 1990. 224 с.

2 Бородин В.Б., Шагурин И.И. Микроконтроллеры. Архитектура, программирование, интерфейс. М.: Издательство ЭКОМ, 1999. 400 с.

3 Emdedded Microcontrollers and Processors., Volume 1. Intel Corp. 1996.

4 Боборыкин А.В., Липовецкий Г.П., Литвинский Г.В. и др. Однокристальные микроЭВМ. М.: МИКАП, 1994. 400 с.

Приложение

СИСТЕМА КОМАНД МИКРОКОНТРОЛЛЕРА INTEL 8051

П.1 Команды передачи данных

Название команды	Мнемокод	КОП	Т	Б	Ц	Операция
Пересылка в аккумулятор из регистра (n = 0…7)	MOV A, Rn	11101rrr	1	1	1	(A) ← (Rn)
Пересылка в аккумулятор прямоадресуемого байта	MOV A, ad	11100101	3	2	1	(A) ← (ad)
Пересылка в аккумулятор байта из РПД (i = 0,1)	MOV A, @Ri	1110011i	1	1	1	(A) ← ((Ri))
Загрузка в аккумулятор константы	MOV A, #d	01110100	2	2	1	(A) ← #d
Пересылка в регистр из аккумулятора	MOV Rn, A	11111rrr	1	1	1	(Rn) ← (A)
Пересылка в регистр прямоадресуемого байта	MOV Rn, ad	10101rrr	3	2	2	(Rn) ← (ad)
Загрузка в регистр константы	MOV Rn, #d	01111rrr	2	2	1	(Rn) ← #d
Пересылка по прямому адресу аккумулятора	MOV ad, A	11110101	3	2	1	(ad) ← (A)
Пересылка по прямому адресу регистра	MOV ad, Rn	10001rrr	3	2	2	(ad) ← (Rn)
Пересылка прямоадресуемого байта по прямому адресу	MOV add, ads	10000101	9	3	2	(add) ← (ads)
Пересылка байта из РПД по прямому адресу	MOV ad, @Ri	1000011i	3	2	2	(ad) ← ((Ri))
Пересылка по прямому адресу константы	MOV ad, #d	01110101	7	3	2	(ad) ← #d
Пересылка в РПД из аккумулятора	MOV @Ri, A	1111011i	1	1	1	((Ri)) ← (A)
Пересылка в РПД прямоадресуемого байта	MOV @Ri, ad	0110011i	3	2	2	((Ri)) ← (ad)
Пересылка в РПД константы	MOV @Ri, #d	0111011i	2	2	1	((Ri)) ← #d
Загрузка указателя данных	MOV DPTR, #d16	10010000	13	3	2	(DPTR) ← #d16

Продолжение прил. П.1

Название команды	Мнемокод	КОП	Т	Б	Ц	Операция
Пересылка в аккумулятор байта из ПП	MOVC A, @A+DPTR	10010011	1	1	2	(A) ← ((A) + (DPTR))
Пересылка в аккумулятор байта из ПП	MOVC A, @A+PC	10000011	1	1	2	(PC) ← (PC)+1, (A) ← ((A)+(PC))
Пересылка в аккумулятор байта из ВПД	MOVX A, @Ri	1110001i	1	1	2	(A) ← ((Ri))

Пересылка в аккумулятор байта из расширенной ВПД	MOVX A, @DPTR	11100000	1	1	2	(A) ← ((DPTR))
Пересылка в ВПД из аккумулятора	MOVX @Ri, A	1111001i	1	1	2	((Ri)) ← (A)
Пересылка в расширенную ВПД из аккумулятора	MOVX @DPTR, A	11110000	1	1	2	((DPTR)) ← (A)
Загрузка в стек	PUSH ad	11000000	3	2	2	(SP) ← (SP) + 1, ((SP)) ← (ad)
Извлечение из стека	POP ad	11010000	3	2	2	(ad) ← (SP), (SP) ← (SP) - 1
Обмен аккумулятора с регистром	XCH A, Rn	11001rrr	1	1	1	(A) ↔ (Rn)
Обмен аккумулятора с прямоадресуемым байтом	XCH A, ad	11000101	3	2	1	(A) ↔ (ad)
Обмен аккумулятора с байтом из РПД	XCH A, @Ri	1100011i	1	1	1	(A) ↔ ((Ri))
Обмен младших тетрад аккумулятора и байта РПД	XCHD A, @Ri	1101011i	1	1	1	$(A_{0...3}) \leftrightarrow ((Ri)_{0...3})$

П.2 Арифметические операции

Название команды	Мнемокод	КОП	Т	Б	Ц	Операция
Сложение аккумулятора с регистром (n = 0…7)	ADD A, Rn	00l01rrr	1	1	1	(A) ← (A) + (Rn)
Сложение аккумулятора с прямоадресуемым байтом	ADD A, ad	00100101	3	2	1	(A) ← (A) + (ad)
Сложение аккумулятора с байтом из РПД (i = 0,1)	ADD A, @Ri	0010011i	1	1	1	(A) ← (A) + ((Ri))
Сложение аккумулятора с константой	ADD A, #d	00100100	2	2	1	(A) ← (A) + #d
Сложение аккумулятора с регистром и переносом	ADDC A, Rn	00111rrr	1	1	1	(A) ← (A) + (Rn) + (C)
Сложение аккумулятора с прямоадресуемым байтом и переносом	ADDC A, ad	00110101	3	2	1	(A) ← (A) + (ad) + (C)
Сложение аккумулятора с байтом из	ADDC A, @Ri	001101	1	1	1	(A) ← (A) + ((Ri))

РПД и переносом		1i				+ (C)
Сложение аккумулятора с константой и переносом	ADDC A, #d	00110100	2	2	1	(A) ← (A) + # d + (C)
Десятичная коррекция аккумулятора	DA A	11010100	1	1	1	Если ($A_{0...3}$)>9 или ((AC) = 1), то ($A_{0...3}$) ← ($A_{0...3}$) + 6, затем если ($A_{4...7}$) >9 или ((C) = 1), то ($A_{4...7}$) ← ($A_{4...7}$) + 6

Продолжение прил. П.2

Название команды	Мнемокод	КОП	Т	Б	Ц	Операция
Вычитание из аккумулятора регистра и заема	SUBB A, Rn	10011rrr	1	1	1	(A) ← (A) - (C) - (Rn)
Вычитание из аккумулятора прямоадресуемого байта и заема	SUBB A, ad	10010101	3	2	1	(A) ← (A) - (C) - ((ad))
Вычитание из аккумулятора байта РПД и заема	SUBB A, @Ri	10010111i	1	1	1	(A) ← (A) - (C) - ((Ri))
Вычитание из аккумулятора константы и заема	SUBB A, d	10010100	2	2	1	(A) ← (A) - (C) - #d
Инкремент аккумулятора	INC A	00000100	1	1	1	(A) ← (A) + 1
Инкремент регистра	INC Rn	00001rrr	1	1	1	(Rn) ← (Rn) + 1
Инкремент прямоадресуемого байта	INC ad	00000101	3	2	1	(ad) ← (ad) + 1
Инкремент байта в РПД	INC @Ri	00000111i	1	1	1	((Ri)) ← ((Ri)) + 1
Инкремент указателя данных	INC DPTR	10100011	1	1	2	(DPTR) ← (DPTR) + 1
Декремент аккумулятора	DEC A	00010100	1	1	1	(A) ← (A) – 1
Декремент регистра	DEC Rn	00011rrr	1	1	1	(Rn) ← (Rn) – 1
Декремент прямоадресуемого байта	DEC ad	00010101	3	2	1	(ad) ← (ad) – 1
Декремент байта в РПД	DEC @Ri	00010111i	1	1	1	((Ri)) ← ((Ri)) - 1
Умножение аккумулятора на регистр B	MUL AB	10100100	1	1	4	(B)(A) ← (A)*(B)
Деление аккумулятора на регистр B	DIV AB	10000100	1	1	4	(B).(A) ← (A)/(B)

П.3 Логические операции

Название команды	Мнемокод	КОП	Т	Б	Ц	Операция
Логическое И аккумулятора и регистра	ANL A, Rn	01011rrr	1	1	1	(A) ← (A) AND (Rn)
Логическое И аккумулятора и прямоадресуемого байта	ANL A, ad	01010101	3	2	1	(A) ← (A) AND (ad)
Логическое И аккумулятора и байта из РПД	ANL A, @Ri	0101011i	1	1	1	(A) ← (A) AND ((Ri))
Логическое И аккумулятора и константы	ANL A, #d	01010100	2	2	1	(A) ← (A) AND #d
Логическое И прямоадресуемого байта и аккумулятора	ANL ad, A	01010010	3	2	1	(ad) ← (ad) AND (A)
Логическое И прямоадресуемого байта и константы	ANL ad, #d	01010011	7	3	2	(ad) ← (ad) AND #d
Логическое ИЛИ аккумулятора и регистра	ORL A, Rn	01001rrr	1	1	1	(A) ← (A) OR (Rn)
Логическое ИЛИ аккумулятора и прямоадресуемого байта	ORL A, ad	01000101	3	2	1	(A) ← (A) OR (ad)
Логическое ИЛИ аккумулятора и байта из РПД	ORL A, @Ri	0100011i	1	1	1	(A) ← (A) OR ((Ri))
Логическое ИЛИ аккумулятора и константы	ORL A, #d	01000100	2	2	1	(A) ← (A) OR #d
Логическое ИЛИ прямоадресуемого байта и аккумулятора	ORL ad, A	01000010	3	2	1	(ad) ← (ad) OR (A)
Логическое ИЛИ прямоадресуемого байта и константы	ORL ad, #d	01000011	7	3	2	(ad) ← (ad) OR #d
Исключающее ИЛИ аккумулятора и регистра	XRL A, Rn	01101rrr	1	1	1	(A) ← (A) XOR (Rn)
Исключающее ИЛИ аккумулятора и прямоадресуемого байта	XRL A, ad	01100101	3	2	1	(A) ← (A) XOR (ad)

Продолжение прил. П.3

Название команды	Мнемокод	КОП	Т	Б	Ц	Операция
Исключающее ИЛИ аккумулятора и байта из РПД	XRL A, @Ri	0110011i	1	1	1	(A) ← (A) XOR

						((Ri))
Исключающее ИЛИ аккумулятора и константы	XRL A, #d	01100100	2	2	1	(A) ← (A) XOR #d
Исключающее ИЛИ прямоадресуемого байта и аккумулятора	XRL ad, A	01100010	3	2	1	(ad) ← (ad) XOR (A)
Исключающее ИЛИ прямоадресуемого байта и константы	XRL ad, #d	01100011	7	3	2	(ad) ← (ad) XOR #d
Сброс аккумулятора	CLR A	11100100	1	1	1	(A) ← 0
Инверсия аккумулятора	CPL A	11110100	1	1	1	(A) ← NOT(A)
Сдвиг аккумулятора влево циклический	RL A	00100011	1	1	1	$(A_{n+1}) \leftarrow (A_n)$, n = 0÷6, $(A_0) \leftarrow (A_7)$
Сдвиг аккумулятора влево через перенос	RLC A	00110011	1	1	1	$(A_{n+1}) \leftarrow (A_n)$, n=0÷6 $(A_0) \leftarrow (C)$, $(C) \leftarrow (A_7)$
Сдвиг аккумулятора вправо цикличе-ский	RR A	00000011	1	1	1	$(A_n) \leftarrow (A_{n+1})$, n = 0÷6, $(A_7) \leftarrow (A_0)$
Сдвиг аккумулятора вправо через пере-нос	RRC A	00010011	1	1	1	$(A_n) \leftarrow (A_{n+1})$, n = 0÷6 $(A_7) \leftarrow (C)$, $(C) \leftarrow (A_0)$
Обмен местами тетрад в аккумуляторе	SWAP A	11000100	1	1	1	$(A_{0\ldots3}) \leftrightarrow (A_{4\ldots7})$

П.1.4 Команды передачи управления

Название команды	Мнемокод	КОП	Т	Б	Ц	Операция
Длинный переход в полном объеме ПП	LJMP ad16	00000010	12	3	2	(PC) ← ad16

Название команды	Мнемокод	КОП	Т	Б	Ц	Операция
Абсолютный переход внутри страницы в 2 Кб	AJMP ad11	$a_{10}a_9a_8$00001	6	2	2	(PC) ← (PC) + 2, ($PC_{0\text{-}10}$) ← ad11
Короткий относительный переход внутри страницы в 256 байт	SJMP rel	10000000	5	2	2	(PC) ← (PC) + 2, (PC) ← (PC) + rel
Косвенный относительный переход	JMP @A+DPTR	01110011	1	1	2	(PC) ← (A) + (DPTR)
Переход, если аккумулятор равен нулю	JZ rel	01100000	5	2	2	(PC)←(PC)+2, если (A) = 0, то (PC)←(PC)+rel
Переход, если аккумулятор не равен нулю	JNZ rel	01110000	5	2	2	(PC)←(PC)+2, если (A)≠0, то (PC)←(PC)+rel
Переход, если перенос равен единице	JC rel	01000000	5	2	2	(PC)←(PC)+2, если (C) = 1, то (PC)←(PC)+rel
Переход, если перенос равен нулю	JNC rel	01010000	5	2	2	(PC)←(PC)+2, если (C) = 0, то (PC)←(PC)+rel

Продолжение прил. П.4

Название команды	Мнемокод	КОП	Т	Б	Ц	Операция
Переход, если бит равен единице	JB bit, rel	00100000	11	3	2	(PC)←(PC)+3, если (b) = 1, то (PC)←(PC)+rel
Переход, если бит равен нулю	JNB bit, rel	00110000	11	3	2	(PC)←(PC)+3, если (b) = 0, то (PC)←(PC)+rel
Переход, если бит установлен, с последующим сбросом бита	JBC bit, rel	00010000	11	3	2	(PC) ← (PC) + 3, если (b) = 1, то (b) ← 0 и (PC)← (PC) + rel
Декремент регистра и переход, если не нуль	DJNZ Rn, rel	11011rrr	5	2	2	(PC) ← (PC) + 2, (Rn) ← (Rn) - 1, если (Rn) ≠ 0, то (PC) ← (PC) + rel
Декремент прямоадресуемого байта и переход, если не нуль	DJNZ ad, rel	11010101	8	3	2	(PC) ← (PC) + 2, (ad) ← (ad) - 1, если (ad) ≠ 0, то (PC) ← (PC) + rel
Сравнение аккумулятора с прямоадресуемым байтом и переход, если не равно	CJNE A, ad, rel	10110101	8	3	2	(PC) ← (PC) + 3, если (A) ≠ (ad), то (PC) ← (PC) + rel, если (A) < (ad), то (C) ← 1, иначе (C) ← 0
Сравнение аккумулятора с константой и переход, если не равно	CJNE A, #d, rel	10110100	10	3	2	(PC) ← (PC) + 3, если (A) ≠ #d, то (PC) ← (PC) + rel,

Название команды	Мнемокод	КОП	Т	Б	Ц	Операция
						если (A) < #d, то (C) ← 1, иначе (C) ← 0

Продолжение прил. П.4

Название команды	Мнемокод	КОП	Т	Б	Ц	Операция
Сравнение регистра с константой и переход, если не равно	CJNE Rn, #d, rel	10111rrr	10	3	2	(PC) ← (PC) + 3, если (Rn) ≠ #d, то (PC) ← (PC) + rel, если (Rn) < #d, то (C) ← 1, иначе (C) ← 0
Сравнение байта в РПД с константой и переход, если не равно	CJNE @Ri, d, rel	1011011i	10	3	2	(PC) ← (PC) + 3, если ((Ri)) ≠ #d, то (PC) ← (PC) + rel, если ((Ri)) < #d, то (C) ← 1, иначе © ← 0
Длинный вызов подпрограммы	LCALL adl6	00010010	12	3	2	(PC) ← (PC) + 3, (SP) ← (SP) + 1, ((SP)) ← ($PC_{0...7}$), (SP) ← (SP) + 1, ((SP)) ← ($PC_{8...15}$), (PC) ← ad16
Абсолютный вызов подпрограммы в пределах страницы в 2 Кб	ACALL ad11	$a_{10}a_9a_8$10001	6	2	2	(PC) ← (PC) + 2, (SP) ← (SP) + 1, ((SP)) ← ($PC_{0...7}$), (SP) ← (SP) + 1, ((SP)) ← ($PC_{8...15}$), ($PC_{0\text{-}10}$) ←

Название команды	Мнемокод	КОП	Т	Б	Ц	Операция
						ad11

Окончание прил. П.4

Название команды	Мнемокод	КОП	Т	Б	Ц	Операция
Возврат из подпрограммы	RET	00100010	1	1	2	$(PC_{8...15}) \leftarrow ((SP))$, $(SP) \leftarrow (SP) - 1$, $(PC_{0...7}) \leftarrow ((SP))$, $(SP) \leftarrow (SP) - 1$
Возврат из подпрограммы обработки прерывания	RETI	00110010	1	1	2	$(PC_{8...15}) \leftarrow ((SP))$, $(SP) \leftarrow (SP) - 1$, $(PC_{0...7}) \leftarrow ((SP))$, $(SP) \leftarrow (SP) - 1$
Пустая операция	NOP	00000000	1	1	1	$(PC) \leftarrow (PC) + 1$

П р и м е ч а н и е. Ассемблер допускает использование обобщенного имени команд JMP и CALL, которые в процессе трансляции заменяются оптимальными по формату командами перехода (AJMP, SJMP, LJMP) или вызова (ACALL, LCALL).

П.5 Операции с битами

Название команды	Мнемо-код	КОП	Т	Б	Ц	Операция
Сброс переноса	CLR C	11000011	1	1	1	(C) ← 0
Сброс бита	CLR bit	11000010	4	2	1	(b) ← 0
Установка переноса	SETB C	11010011	1	1	1	(C) ← 1
Установка бита	SETB bit	11010010	4	2	1	(b) ← 1
Инверсия переноса	CPL C	10110011	1	1	1	(C) ← NOT(C)
Инверсия бита	CPL bit	10110010	4	2	1	(b) ← NOT(b)
Логическое И бита и переноса	ANL C, bit	10000010	4	2	2	(C) ← (C) AND (b)
Логическое И инверсии бита и переноса	ANL C, /bit	10110000	4	2	2	(C) ← (C) AND (NOT(b))
Логическое ИЛИ бита и переноса	ORL C, bit	01110010	4	2	2	(C) ← (C) OR (b)
Логическое ИЛИ инверсии бита и переноса	ORL C, /bit	10100000	4	2	2	(C) ← (C) OR (NOT(b))
Пересылка бита в перенос	MOV C, bit	10100010	4	2	1	(C) ← (b)
Пересылка переноса в бит	MOV bit, C	10010010	4	2	2	(b) ← (C)