Система команд микроконтроллера семейства 8051

Система команд микроконтроллера включает 111 основных команд длиной от 1 до 3 байт, большинство из них - одно- или двухбайтные. Все команды выполняются за один или два машинных цикла (соответственно 1 или 2 мкс при тактовой частоте 12 МГц), кроме умножения и деления, выполняемых за четыре машинных цикла (4 мкс). Все машинные циклы состоят из 6 состояний S1…S6, каждое из которых содержит две фазы P1 и P2. Микроконтроллеры используют регистровую, прямую, непосредственную, косвенную, индексную и стековую адресацию данных. Операндами могут быть отдельные биты, четырехбитные цифры, байты и двухбайтные слова.

Типы (форматы) и группы команд

Всего микроконтроллеры выполняют 13 типов команд, приведенных в таблице. Как следует из нее, первый байт команды всегда содержит код операции (КОП), а второй и третий (если они присутствуют в команде) - адреса операндов или их непосредственные значения.

Тип команды	Первый байт D7...D0	Второй байт D7...D0	Третий байт D7...D0
тип 1	коп		
тип 2	коп	#d	
тип 3	коп	ad	
тип 4	коп	bit	
тип 5	коп	rel	
тип 6	коп	a7...a0	
тип 7	коп	ad	#d
тип 8	коп	ad	rel
тип 9	коп	ads	add
тип 10	коп	#d	rel
тип 11	коп	bit	rel
тип 12	коп	ad16h	ad16l
тип 13	коп	#d16h	#d16l

Команды делятся на 5 функциональных групп: пересылки данных, арифметических операций, логических операций, операций над битами, передачи управления.

Обозначения, используемые при описании команд

Rn (n = 0, 1,..., 7) - регистр общего назначения в выбранном банке регистров;
@Ri (i = 0, 1) - регистр общего назначения в выбранном банке регистров, используемый в качестве регистра косвенного адреса;
ad - адрес прямо адресуемого байта;
ads - адрес прямо адресуемого байта-источника;
add - адрес прямо адресуемого байта-получателя;
ad11 - 11-разрядный абсолютный адрес перехода;
ad16 - 16-разрядный абсолютный адрес перехода;
rel - относительный адрес перехода;
#d - непосредственный операнд;
#d16 - непосредственный операнд (2 байта);
bit - адрес прямо адресуемого бита;
/bit - инверсия прямо адресуемого бита;
A - аккумулятор;
PC - счетчик команд;
DPTR - регистр указатель данных;
( ) - содержимое ячейки памяти или регистра,

Команды пересылки данных

Эта группа представлена 28 командами, представленных в таблице, где указаны код операции (КОП), тип команды (Т), ее длина в байтах (В), время выполнения в машинных циклах (С) и краткое описание.

Мнемокод	КОП	Т	В	С	Описание
MOV A, Rn	11101rrr	1	1	1	(A) <-- (Rn)
MOV A, ad	11100101	3	2	1	(A) <--(ad)
MOV A, @Ri	1110011i	1	1	1	(A) <-- ((Ri))
MOV A, #d	01110100	2	2	1	(A) <-- #d
MOV Rn, A	11111rrr	1	1	1	(Rn) <-- (A)
MOV Rn, ad	10101rrr	3	2	2	(Rn) <-- (ad)
MOV Rn, #d	01111rrr	2	2	1	(Rn) <-- #d
MOV ad, A	11110101	3	2	1	(ad) <--(A)
MOV ad, Rn	10001rrr	3	2	2	(ad) <-- (Rn)
MOV add, ads	10000101	9	3	2	(add) <-- (ads)
MOV ad, @Ri	1000011i	3	2	2	(ad) <-- ((Ri))
MOV ad, #d	01110101	7	3	2	(ad) <-- #d
MOV @Ri, A	1111011i	1	1	1	((Ri)) <-- (A)
MOV @Ri, ad	0110011i	3	2	2	((Ri)) <-- (ad)
MOV @Ri, #d	0111011i	2	2	1	((Ri)) <-- #d
MOV DPTR, #d16	10010000	3	3	2	(DPTR) <-- #d16
MOVC A, @A+DPTR	10010011	1	1	2	(A) <-- ((A)+(DPTR))
MOVC A, @A+PC	10000011	4	1	2	(PC) <-- (PC+1), (A) <-- ((A)+(PC))
MOVX A, @Ri	11100011	1	1	2	(A) <-- ((Ri))
MOVX A, @DPTR	11100000	1	1	2	(A) <-- ((DPTR))
MOVX @Ri, A	1111001i	1	1	2	((Ri)) <-- (A)
MOVX @DPTR, A	11110000	1	1	2	((DPTR)) <-- (A)
PUSH ad	11000000	3	2	2	(SP) <-- (SP)+1, ((SP)) <-- (ad)
POP ad	11010000	3	2	2	(ad) <-- ((SP)), (SP) <-- (SP)-1
XCH A, Rn	11001rrr	1	1	1	(A) <-> (Rn)
XCH A, ad	11000101	3	2	1	(A) <-->(ad)
XCH A, @Ri	1100011i	1	1	1	(A) <->((@Ri))
XCHD A, @Ri	1101011i	1	1	1	(A_{0-3})<->$((@Ri))_{0-3}$

По команде MOV выполняется пересылка данных из второго операнда в первый. Для доступа к внешней памяти данных, ни к памяти программ предназначены команды M0VX и MOVC соответственно. По команде XCH выполняется обмен байтами между аккумулятором и ячейкой РПД, а по команде XCHD - обмен младшими тетрадами (битами 0 - 3).

Команды PUSH и POP предназначены для записи данных в стек и их чтения из стека, создаваемого во внутреннем ОЗУ. В процессе инициализации микроконтроллера после сигнала сброса или при включении питающего напряжения в SP заносится код 07H. Это означает, что первый элемент стека будет располагаться в ячейке памяти с адресом 08H.

Доступ к регистру PSW, таймеру, портам ввода-вывода и к другим регистрам специальных функций, осуществляется заданием соответствующего прямого адреса, т.е. это команды обычных пересылок, в которых вместо адреса можно ставить название соответствующего регистра. Например, чтение PSW в аккумулятор может быть выполнено командой MOV A, PSW, которая эквивалентна команде MOV A, 0D0h , где D0 - адрес PSW.

Кроме того, следует отметить, что аккумулятор имеет два различных имени в зависимости от способа адресации: A - при регистровой адресации (например, MOV A, R0) и ACC - при использовании прямого адреса.

Команды арифметических операций

В данную группу входят 24 команды арифметических операций с целочисленными данными, включая команды умножения и деления.

Мнемокод	КОП	T	B	C	Описание
ADD A, Rn	00101rrr	1	1	1	(A)<-(A)+(Rn)
ADD A, ad	00100101	3	2	1	(A)<-(A)+(ad)
ADD A, @Ri	0010011i	1	1	1	(A)<-(A)+((Ri))
ADD A, #d	00100100	2	2	1	(A)<-(A)+#d
ADDC A, Rn	00111rrr	1	1	1	(A)<-(A)+(Rn)+(C)
ADDC A, ad	00110101	3	2	1	(A)<-(A)+(ad)+(C)
ADDC A, @Ri	0011011i	1	1	1	(A)<-(A)+((Ri))+C)
ADDC A, #d	00110100	2	2	1	(A)<-(A)+#d+(C)
DAA	11010100	1	1	1	Десятичная коррекция аккумулятора
SUBB A, Rn	10011rrr	1	1	1	(A)<-(A)-(Rn)-(C)
SUBB A, ad	10010101	3	2	1	(A)<-(A)-(ad)-(C)
SUBB A, @Ri	1001011i	1	1	1	(A)<-(A)-((Ri))-(C)
SUBB A, #d	10010100	2	2	1	(A)<-(A)-#d-(C)
INC A	00000100	1	1	1	(A)<-(A)+1
INC Rn	00001rrr	1	1	1	(Rn)<-(Rn)+1
INC ad	00000101	3	2	1	(ad)<-(ad)+1
INC @Ri	0000011i	1	1	1	((Ri)<-((Ri))+1
INC DPTR	10100011	1	1	2	(DPTR)<-(DPTR)+1
DEC A	00010100	1	1	1	(A)<-(A)-1
DEC Rn	00011rrr	1	1	1	(Rn)<-(Rn)-1
DEC ad	00010101	3	2	1	(ad)<-(ad)-1
DEC @Ri	0001011i	1	1	1	((Ri))<-((Ri))-1
MUL AB	10100100	1	1	4	(B)(A)<-(A)*(B)
DIV AB	10000100	1	1	4	(A).(B)<-(A)/(B)

По результату выполнения команд ADD, ADDC, SUBB, MUL и DIV устанавливаются флаги CY, AC, OV и P регистра PSW, структура которого приведена в следующей таблице.

flag	Адрес флага	Описание
CY	0D7h	Перенос из 7 (старшего) разряда
AC	0D6h	Перенос из 3 разряда
F0	0D5h	Произвольно используется программистом
RS1	0D4h	Старший разряд номера банка
RS0	0D3h	Младший разряд номера банка
OV	0D2h	Переполнение результата
	0D1h	Безымянный, используется программистом
P	0D0h	Признак четного количества единиц результата

Флаг CY устанавливается при переносе из разряда D7, т. е. в случае, если результат не помещается в восемь разрядов; флаг AC устанавливается при переносе из разряда D3 в командах сложения и вычитания и служит для реализации десятичной арифметики. Этот признак используется командой DAA.

Флаг OV устанавливается при переносе из разряда D6, т. е. в случае, если результат не помещается в семь разрядов и восьмой не может быть интерпретирован как знаковый. Этот признак служит для организации обработки чисел со знаком.

Наконец, флаг P устанавливается и сбрасывается аппаратно. Если число единичных бит в аккумуляторе нечетно, то P = 1, в противном случае P = 0.

Команды логических операций

В этой группе 25 команд. Они позволяют выполнять операции над байтами: логическое И ($\wedge$), логическое ИЛИ ($\vee$), исключающее ИЛИ ($\oplus$), инверсию (NOT), сброс в 0 и сдвиг.

Мнемокод	КОП	Т В С	Описание
ANL A, Rn	01011rrr	1 1 1	(A) <- (A) $\wedge$ n)
ANL A, ad	01010101	3 2 1	(A) <- (A) $\wedge$ (ad)
ANL A, @Ri	01010111	1 1 1	(A) <- (A) $\wedge$ ((Ri))
ANL A, #d	01010100	2 2 1	(A) <- (A) $\wedge$ #d
ANL ad, A	01010010	3 2 1	(ad) <- (ad) $\wedge$ (A)
ANL ad, #d	01010011	7 3 2	(ad) <- (ad) $\wedge$ #d
ORL A, Rn	01001rrr	1 1 1	(A) <- (A) $\vee$ (Rn)
ORL A, ad	01000101	3 2 1	(A) <- (A) $\vee$ (ad)
ORL A, @Ri	0100011i	1 1 1	(A) <- (A) $\vee$ ((Ri)
ORL A, #d	01000100	2 2 1	(A) <- (A) $\vee$ #d
ORL ad, A	01000010	3 2 1	(ad) <- (ad) $\vee$ A
ORL ad, #d	01000011	7 3 2	(ad) <- (ad) $\vee$ #d
XRL A, Rn	01101rrr	1 1 1	(A) <- (A) $\oplus$ (Rn)
XRL A, ad	01100101	3 2 1	(A) <- (A) $\oplus$ (ad)
XRL A, @Ri	0110011i	1 1 1	(A) <- (A) $\oplus$ ((Ri))
XRL A, #d	01100100	2 2 1	(A) <- (A) $\oplus$ #d
XRL ad, A	01100010	3 2 1	(ad) <- (ad) $\oplus$ A
XRL ad, #d	01100011	7 3 2	(ad) <- (ad) $\oplus$ #d
CLR A	11100100	1 1 1	(A) <- 0
CPL A	11110100	1 1 1	(A) <- NOT(A)
SWAP A	11000100	1 1 1	(A_{0-3}) <-> (A_{4-7})
RL A	00100011	1 1 1	Циклический сдвиг влево
RLC A	00110011	1 1 1	Сдвиг влево через перенос
RR A	00000011	1 1 1	Циклический сдвиг вправо
RRC A	00010011	1 1 1	Сдвиг вправо через перенос

Команды операций над битами

Группа состоит из 12 команд, позволяющих выполнять операции над отдельными битами: сброс, установку, инверсию бита, а также логические И и ИЛИ. Одним из операндов в операциях с двумя операндами выступает признак переноса CY, в качестве операндов могут использоваться 128 бит из резидентной памяти данных и регистры специальных функций, допускающие адресацию отдельных бит.

Мнемокод	КОП	Т В С	Описание
CLR C	11000011	1 1 1	(C) <- 0
CLR bit	11000010	4 2 1	(bit) <- 0
SETB C	11010011	1 1 1	(C) <- 1
SETB bit	11010010	4 2 1	(bit) <- 1
CPL C	10110011	1 1 1	(C) <- NOT(C)
CPL bit	10110010	4 2 1	(bit) <- NOT (bit)
ANL C, bit	10000010	4 2 2	(C)<- (C) $\wedge$ (bit)
ANL C, /bit	10110000	4 2 2	(C) <- (C) $\wedge$ NOT(bit)
ORL C, bit	01110010	4 2 2	(C) <- (C) $\vee$ (bit)
ORL C, /bit	10100000	4 2 2	(C) <- (C) $\vee$ NOT(bit)
MOV C, bit	10100010	4 2 1	(C) <- (bit)
MOV bit, C	10010010	4 2 2	(bit) <- (C)

Команды передачи управления

Группа представлена командами безусловного и условного переходов, командами вызова подпрограмм и командами возврата из подпрограмм.

Мнемокод	КОП	Т В С	Описание
LJMP ad16	00000010	12 3 2	Длинный безусловный переход по всей памяти
AJMP ad11	00001	6 2 2	Безусловный переход в пределах страницы 2 Кбайт
SJMP rel	10000000	5 2 2	Безусловный переход в пределах страницы 256 байт
JMP @A+DPTR	01110011	1 1 2	Безусловный переход по косвенному адресу
JZ rel	01100000	5 2 2	Переход, если нуль
JNZ rel	01110000	5 2 2	Переход, если не нуль
JC rel	01000000	5 2 2	Переход, если бит переноса установлен
JNC rel	01010000	5 2 2	Переход, если бит переноса не установлен
JB bit, rel	00100000	11 3 2	Переход, если бит установлен
JNB bit, rel	00110000	11 3 2	Переход, если бит не установлен
JBC bit, rel	00010000	11 3 2	Переход, если бит установлен со сбросом бита
DJNZ Rn, rel	11011rrr	5 2 2	Декремент Rn и переход, если не нуль
DJNZ ad, rel	11010101	8 3 2	Декремент (ad) и переход, если не нуль
CJNE A, ad, rel	10110101	8 3 2	Сравнение аккумулятора с байтом и переход, если не равно
CJNE A, #d, rel	10110100	10 3 2	Сравнение аккумулятора с константой и переход, если неравно
CJNE Rn, #d, rel	10111rrr	10 3 2	Сравнение регистра с константой и переход, если не равно
CJNE @Ri, #d, rel	1011011i	10 3 2	Сравнение байта памяти с константой и переход, если не равно
LCALL ad16	00010010	12 3 2	Длинный вызов подпрограммы во всей памяти
ACALL ad11	10001	6 2 2	Вызов подпрограммы в пределах страницы 2 Кбайт
RET	00100010	1 1 2	Возврат подпрограммы
RETI	00110010	1 1 2	Возврат подпрограммы обработки прерывания
NOP	00000000	1 1 1	Пустая операция

Команда безусловного перехода LJMP (L - long - длинный) осуществляет переход по абсолютному 16-битному адресу, указанному в теле команды, т. е. команда обеспечивает переход в любую точку памяти программ.

Действие команды AJMP (A - absolute - абсолютный) аналогично команде LJMP, однако в теле команды указаны лишь 11 младших разрядов адреса. Поэтому переход осуществляется в пределах страницы размером 2 Кбайт, при этом надо иметь в виду, что сначала содержимое счетчика команд увеличивается на 2 и только потом заменяются 11 разрядов адреса.

В отличие от предыдущих команд, в команде SJMP (S - short - короткий) указан не абсолютный, а относительный адрес перехода. Величина смещения rel рассматривается как число со знаком, а, следовательно, переход возможен в пределах -128...+127 байт относительно адреса команды, следующей за командой SJMP.

Команда косвенного перехода JMP @A+DPTR позволяет вычислять адрес перехода в процессе выполнения самой программы.

Командами условного перехода можно проверять следующие условия:

- JZ - аккумулятор содержит нулевое значение;
- JNZ - аккумулятор содержит не нулевое значение
- JC - бит переноса C установлен;
- JNC - бит переноса C не установлен;
- JB - прямо адресуемый бит равен 1
- JNB - прямо адресуемый бит равен 0;
- JBC - прямо адресуемый бит равен 1 и сбрасывается в нулевое значение при выполнении команды.

Все команды условного перехода рассматриваемых микроконтроллеров содержат короткий относительный адрес, т. е. переход может осуществляться в пределах -128... +127 байт относительно следующей команды.

Команда DJNZ предназначена для организации программных циклов. Регистр Rn или байт по адресу ad, указанные в теле команды, содержат счетчик повторений цикла, а смещение rel - относительный адрес перехода к началу цикла. При выполнении команды содержимое счетчика уменьшается на 1 и проверяется на 0. Если значение содержимого счетчика не равно 0, то осуществляется переход на начало цикла, в противном случае выполняется следующая команда.

Команда CJNE удобна для реализации процедур ожидания внешних событий. В теле команды указаны "координаты" двух байт и относительный адрес перехода rel. В качестве двух байт могут быть использованы, например, значения содержимого аккумулятора и прямо адресуемого байта или косвенно адресуемого байта и константы. При выполнении команды значения указанных двух байт сравниваются и в случае, если они не одинаковы, осуществляется переход. Например, команда WAIT: CJNE A, P0, WAIT будет выполняться до тех пор, пока значения на линиях порта P0 не совпадут со значениями содержимого аккумулятора.

Действие команд вызова процедур полностью аналогично действию команд безусловного перехода. Единственное отличие состоит в том, что они сохраняют в стеке адрес возврата.

Команда возврата из подпрограммы RET восстанавливает из стека значение содержимого счетчика команд, а команда возврата из процедуры обработки прерывания RETI, кроме того, разрешает прерывание обслуженного уровня. Команды RET и RETI не различают, какой командой - LCALL или ACALL - была вызвана подпрограмма, так как и в том, и в другом случае в стеке сохраняется полный 16-разрядный адрес возврата.

Большинство Ассемблеров допускают обобщенную мнемонику JMP - для команд безусловного перехода и CALL - для команд вызова подпрограмм. Конкретный тип команды определяется Ассемблером, исходя из "длины"перехода или вызова,

Директивы ассемблера

Директивы ассемблера служат для управления процессом трансляции.

.ORG *адрес* - задание адреса размещения последующего фрагмента программы в памяти
.END - конец исходного текста программы
.CHIP *тип* – выбор конкретного типа контроллера
.CODE - начало сегмента кода
.DATA - начало сегмента данных
.RSECT - начало регистровой секции
.BSECT - начало битовой секции
.ENDS - конец любого сегмента и секции
.ABSOLUTE - переключение транслятора в абсолютный режим трансляции
.RELATIVE - переключение транслятора в относительный режим трансляции
.EXTERNAL *список* - перечень внешних имен, определенных в других программных модулях
.PUBLIC *список* - перечень имен, которые должны быть «видны» в других модулях
.GLOBAL ON - все символические имена сделать видимыми во всех модулях
.MODULE - начало модуля
.ENDMOD - конец модуля
.DS *размер* - зарезервировать *размер* байт памяти
.RADIX *основание* - изменить основание системы счисления по умолчанию (2, 8, 10, 16)
.DB *список_значений_через_запятую* - инициализация памяти байтами данных
.BLKB *размер, значение* - инициализация памяти массивом одинаковых байт
.ASCII *строка* – инициализация памяти строкой символов
.DC *строка* – инициализация памяти строкой символов
.DW *список_значений* – инициализация памяти двухбайтовыми словами данных
.BLKW *размер, значение* - инициализация памяти массивом одинаковых двухбайтовых слов