Краткий справочник по системе команд микропроцессора К1801ВМ1

Микропроцессор К1801ВМ1 имеет 8 регистров общего назначения (РОН) и регистр слова состояния процессора (РСП). Все регистры 16-разрядные.

Регистры общего назначения R0, R1, R2, R3, R4, R5 предназначены для хранения промежуточных результатов вычислений.

Регистр R6 обозначается в ассемблере "SP" (Stack Pointer - указатель стека) и используется в этом качестве как при вызове подпрограмм, так и при обработке прерываний.

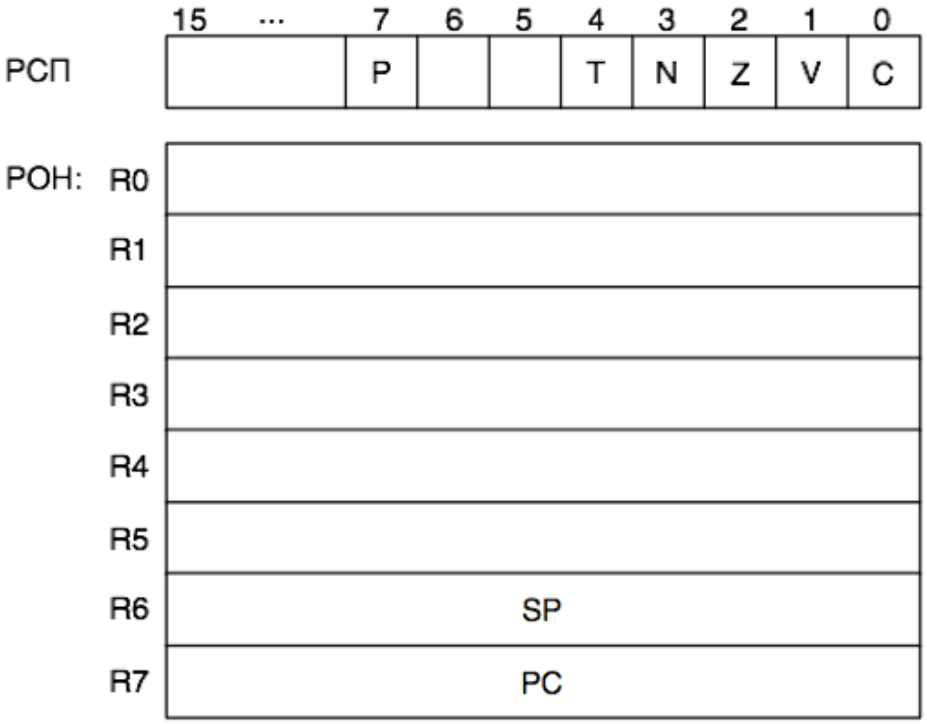
Регистр R7 обозначается "PC" (Program Counter - счетчик команд) всегда содержит адрес следующей команды, котроую должен выполнить процессор.

РСП предназначен для хранения PSW (Processor Status Word - слово состояния процессора). В PSW имеют значение следующие биты (разряды):

бит 0 (C) устанавливается в 1, если при выполнении команды произошел перенос единицы из старшего разряда результата;
бит 1 (V) устанавливается в 1, если при выполнении арифметической команды (например, сложения) произошло арифметическое переполнение;
бит 2 (Z) устанавливается в 1, если результат равен нулю;
бит 3 (N) устанавливается в 1, если результат отрицателен. Эти биты PSW используются командами условного перехода

Остальные биты PSW устанавливаются программистом для задания режимов работы процессора.

бит 4 (T), установленный в 1, вызывает после выполнения очередной команды прерывание по вектору 14. Это используется программами - отладчиками для трассировки программы;
бит 7 (P), установленный в 1, запрещает (маскирует) прерывания от внешних устройств (например, клавиатуры). Таким образом, командой установки PSW MTPS #200 можно запретить прерывания от клавиатуры до тех пор, пока не выполнится команда MTPS #0.



Все опкоды представлены в восьмеричной системе счисления.

Условные сокращения в опкодах:

- ***** - признак переменного размера обрабатываемых данных. Если равен нулю, то операция производится над 16-разрядным словом. Если равен единице, то операция производится над байтом;
- **SS** - поле адресации операнда источника;
- **DD** - поле адресации операнда приёмника;
- **NN** - 6-разрядное смещение;
- **XXX** - 8-разрядное смещение;
- **R** - 3-разрядный индекс регистра общего назначения;

Если команда позволяет работать с байтами, то к мнемонике добавляется суффикс **(B)**. Например мнемоника MOV обозначает, что операция будет производиться над 16-разрядным словом, а мнемоника MOVB обозначает, что операция будет производиться над байтом. Фактически этот модификатор присутствует во всех мненониках, в опкоде которых встречается символ *****.

Поля адресации определяют способ получения операнда.

Код	Способ адресации	Описание	Примеры
0	регистровый	регистр содержит операнд	CLR R1
1	косвенно-регистровый	регистр содержит адрес операнда	CLR (R1) CLR @R1
2	автоинкрементный	регистр содержит адрес операнда. Содержимое регистра после его использования в качестве адреса увеличивается на 2 (для команд над словами) или на 1 (для байтовых команд)	CLR (R2)+
3	косвенно-автоинкрементный	регистр содержит адрес адреса операнда. Содержимое регистра после его использования в качестве адреса увеличивается на 2	CLR @(R2)+
4	автодекрементный	содержимое регистра уменьшается на 2 (для команд над словами) или на 1 (для байтовых команд) и затем используется как адрес операнда	CLR -(R2)

5	косвенно-автодекрементный	содержимое регистра уменьшается на 2 и используется как адрес адреса операнда	CLR @-(R1)
6	индексный	содержимое регистра складывается с числом, записанным после команды, и полученная сумма используется в качестве адреса операнда	CLR 2(R5) CLR MT(R0)
7	косвенно-индексный	содержимое регистра складывается с числом, записанным после команды и полученная сумма используется в качестве адреса адреса операнда	CLR @22(R1)

Если адресация операнда происходит через регистр PC, то способы адресации другие.

Код	Способ адресации	Описание	Примеры
2	непосредственный	операнд хранится в слове, следующем за командой	MOV #21,R3 MOV #IN,R0
3	абсолютный	адрес операнда хранится в слове, следующем за командой	CLR @#7000 JMP @#BEN
6	относительный	содержимое PC складывается со словом, записанным в памяти за командой, и полученная сумма используется как адрес операнда	JMP TV CLR 5554
7	косвенно-относительный	Содержимое PC складывается со словом, следующим за командой, полученная сумма используется как адрес адреса операнда	CLR @MET INC @15342

Обозначения косвенной адресации "(R2)" и "@R2" равнозначны.

Однооперандные команды

1. CLR(B)

Опкод: *050DD

Действие: обнуление операнда.

2. COM(B)

Опкод: *051DD

Действие: инвертирование операнда.

Описание: По этой команде образуется инверсный (обратный) код операнда - во всех разрядах операнда нули заменяются единицами, а единицы - нулями. Это действие представляет собой логическую операцию отрицания ("НЕ").

Пример:

```
005121          COM (R1)+
```

3. INC(B)

Опкод: *052DD

Действие: Увеличивает значение операнда на 1.

Пример:

```
005237 010000  INC @#10000
```

Описание примера:

Команда 005237 увеличивает на 1 содержимое слова памяти с адресом 10000 - на 1 увеличивается число, хранящееся в 2-х ячейках с адресами 10000 и 10001. Две последние цифры "37" в коде команды указывают, что адрес операнда хранится в памяти за кодом команды (абсолютная адресация).

4. DEC(B)

Опкод: *053DD

Действие: Уменьшает значение операнда на 1.

Пример:

```
005312        DEC (R2)
```

Описание примера:

Команда уменьшает на 1 слово, адрес которого хранится в регистре R2.

5. NEG(B)

Опкод: *054DD

Действие: Изменяет знак операнда.

Описание: Операция изменения знака числа эквивалентна получению дополнительного кода числа (то есть числа, которое, будучи сложено с исходным, даст в сумме ноль). Формирование дополнительного двоичного кода числа состоит из двух последовательных операций : получения инверсного (обратного) двоичного кода числа и прибавления 1 (к полученному инверсному коду).

Пример:

```
005412        NEG @R2
```

Описание примера:

Команда изменяет знак числа, хранящегося по адресу, указанному в R2. Если число было положительным, то станет отрицательным, и наоборот.

6. ADC(B)

Опкод: *055DD

Действие: Увеличивает значение операнда на содержимое разряда C PSW.

7. SBC(B)

Опкод: *056DD

Действие: Уменьшает значение операнда на содержимое разряда C PSW.

8. TST(B)

Опкод: *057DD

Действие: Проверка значения операнда.

Описание: По этой команде производится тестирование (проверка) значения операнда и установка в "1" (или сброс в "0") разрядов Z и N PSW, а разряды V и C при этом сбрасываются в "0". Если значение операнда отрицательно, то разряд N PSW установится в "1", иначе сбросится в "0". Если значение операнда нулевое, то разряд Z установится в "1", иначе сбросится в "0". Значение операнда при этом не изменяется.

9. ROR(B)

Опкод: *060DD

Действие: циклический сдвиг вправо.

Описание: Производит циклический сдвиг значений разрядов операнда вправо на один разряд. Значение 15-го разряда загружается в 14-ый разряд, значение 14-го разряда - в 13-ый разряд и так далее. Значение разряда C PSW загружается в 15-ый разряд операнда, а значение нулевого (младшего) разряда операнда - в разряд C PSW.

10. ROL(B)

Опкод: *061DD

Действие: циклический сдвиг влево

Описание: Команда выполняется так же, как и предыдущая, но сдвиг выполняется влево и значение разряда C PSW загружается в нулевой разряд операнда, а значение 15-ого разряда операнда - в разряд C PSW.

11. ASR(B)

Опкод: *062DD

Действие: арифметический сдвиг вправо

Описание: Производит арифметический сдвиг вправо. При этом значение каждого разряда операнда сдвигается на один разряд вправо. Значение нулевого разряда операнда загружается в разряд C PSW. В 14-ый и 15-ый разряды записывается значение 15-ого разряда операнда

12. ASL(B)

Опкод: *063DD

Действие: арифметический сдвиг влево

Описание: Производит арифметический сдвиг значения каждого разряда операнда на один разряд влево. Нулевой разряд операнда очищается, а значение 15-ого разряда операнда загружается в разряд C PSW.

13. SWAB

Опкод: 0003DD

Описание: Меняет местами старший и младший байты операнда (слова).

14. MTPS

Опкод: 1064SS

Описание: Команда записывает в РСП новое значение слова состояния процессора (PSW), равное значению операнда.

Пример:

```
106427 000200  MTPS #200
```

Описание примера:

В данном примере в РСП загружается новое значение PSW, равное 200. Таким образом, устанавливается в 1 бит приоритета "P" в PSW, что запрещает процессору обрабатывать прерывания от клавиатуры до тех пор, пока не будет выполнена команда MTPS #0.

15. MFPS

Опкод: 1067DD

Описание: Команда пересылает PSW в место, определяемое полем адресации операнда.

Пример:

```
106702          MFPS R2
```

Описание примера:

Производится пересылка PSW в регистр R2.

Двухоперандные команды

16. MOV(B)

Опкод: *1SSDD

Действие: копирование значение из одного операнда в другой.

Описание: По этой команде операнд, местонахождение которого определяется полем адресации "SS", пересылается по адресу, определяемому полем адресации "DD". При этом содержимое источника, откуда берется операнд для пересылки, не изменяется. При выполнении байтовой команды MOVB с использованием регистрового способа адресации (для операнда приемника) все разряды старшего байта операнда приемника устанавливаются в "1", если знаковый разряд (старший разряд) младшего байта установлен в "1", иначе разряды старшего байта сбрасываются в "0".

Пример:

```
010204          MOV R2,R4
```

Описание примера:

Копия содержимого регистра R2 пересылается (загружается) в регистр R4. Содержимое R2 при этом не изменяется.

Более сложный пример:

```
012737 177777 070000  MOV #177777,@#70000
```

Описание примера:

Здесь одна команда занимает 3 слова памяти. В 1-ом слове записан код самой команды - число 012737. Во 2-ом слове хранится операнд источника (число 177777), предназначенный для пересылки, и в 3-ем слове хранится адрес приемника 70000, куда пересылается число 177777.

Две цифры "27" в коде команды, записанные в поле адресации операнда источника, указывают, что операнд источника (число 177777) хранится в памяти сразу же за кодом команды. Две цифры "37", записанные в поле адресации операнда приемника указывают, что значение адреса приемника хранится в памяти также за кодом команды - но в данном случае место в памяти сразу же за кодом команды занято числом 177777, поэтому значение адреса приемника (число 70000) записывается в памяти за числом 177777.

17. CMP(B)

Опкод: *2SSDD

Действие: сравнение операндов.

Описание:

Данная команда вычитает из операнда источника операнд приемника. Но при этом значения самих операндов не изменяются. Изменяются лишь значения разрядов C, V, Z, N PSW.

Разряд N устанавливается в "1", если результат вычитания - отрицательное число. Разряд Z устанавливается в "1", если результат - равен нулю (операнды равны). Разряд V устанавливается в "1", если было арифметическое переполнение - если операнды были противоположного знака, а знак результата совпадает со знаком операнда приемника. Разряд C обнуляется, если был перенос из старшего разряда результата вычитания.

Пример:

```
020103        CMP R1,R3
```

Описание примера:

Команда 020103 производит сравнение содержимого регистра R1 с содержимым регистра R3.

18. BIT(B)

Опкод: *3SSDD

Действие: проверка состояния (значения) отдельных разрядов одного из операндов.

Описание:

Значение каждого разряда результата образуется поразрядным логическим умножением значений соответствующих разрядов операнда источника и операнда приемника. Например, значение 12-ого разряда результата образуется логическим умножением значений 12-ого разряда операнда источника и 12-ого разряда операнда приемника.

Значения операнда источника и операнда приемника при выполнении команды не изменяются. Изменяются лишь значения разрядов V,Z,N PSW. Разряд N устанавливается в "1", если в результате поразрядного логического умножения 2-х операндов получилось число, знаковый (старший) разряд которого установлен в "1". Разряд Z устанавливается в "1", если в результате логического умножения получилось число, все разряды которого сброшены в "0". Разряд V PSW сбрасывается в "0".

Пример:

```
032737 000100 177716  BIT #100,@#177716
```

Описание примера: проверка значения седьмого разряда в 16-разрядном слове, расположенном по адресу 177716.

19. BIC(B)

Опкод: *4SSDD

Действие: сброс (обнуление) разрядов.

Описание:

По этой команде сбрасываются в "0" (обнуляются) разряды операнда приемника, соответствующие установленным в "1" разрядам операнда источника. Остальные разряды операнда приемника остаются без изменений. Значение операнда источника при выполнении команды не изменяется. Действие этой команды эквивалентно выполнению последовательности логических операций: инвертирования (отрицание) операнда источника; логического умножения ("И") результата и операнда приемника; запись результата в операнд приемника.

При выполнении команды BIC значения разрядов V,Z,N PSW изменяются в зависимости от значения результата так же, как при выполнении команды BIT.

20. BIS(B)

Опкод: *5SSDD

Действие: установка в "1" разрядов.

Описание:

По этой команде устанавливаются в "1" разряды операнда приемника, соответствующие установленным в "1" разрядам операнда источника. Остальные разряды операнда приемника остаются без изменений. Значение операнда источника при выполнении команды не меняется. Действие этой команды эквивалентно операции логического сложения ("ИЛИ") над операндами источника и приемника.

При выполнении команды BIS значения разрядов V,Z,N PSW изменяются в зависимости от значения результата так же, как при выполнении команды BIT.

21. ADD

Опкод: 06SSDD

Действие: арифметическое сложение

Описание:

Команда суммирует операнд источника с операндом приемника. Результат сложения записывается по адресу операнда приемника. Значение операнда источника при этом не изменяется. Значение разрядов Z и N PSW изменяются так же, как и для команды CMP. Разряд V PSW устанавливается в "1", если в результате выполнения команды произошло арифметическое переполнение, то есть, если оба операнда были одного знака, а результат сложения получился противоположного знака. В обычной арифметике такого не бывает, а в работе процессора такой парадокс возможен в силу того, что разрядность процессора ограничена. Разряд C PSW устанавливается в "1", если был перенос из старшего разряда результата.

Пример:

```
060204          ADD R2,R4
```

Описание примера:

Производится сложение содержимого 2-х регистров: R2 и R4. Результат сложения помещается в регистр R4.

22. SUB

Опкод: 16SSDD

Действие: арифметическое вычитание

Описание:

По этой команде из операнда приемника вычитается операнд источника. Результат помещается по адресу операнда приемника. Изменение разрядов C,Z,N PSW происходит так же, как при выполнении команды CMP. Арифметическое переполнение при выполнении данной команды происходит, когда операнды имели разные знаки, а знак результата совпадает со знаком операнда источника.

Пример:

```
20000: 166767 000004 000006  SUB VR1,S
....
20010: 000056    VR1:  .#56
20012: 000012    VR2:  .#12
20014: 000010      S:  .#10
```

Описание примера:

Команда SUB, хранящаяся по адресу 20000, производит вычитание значения переменной VR1 из значения переменной S. Результат вычитания присваивается переменной S.

По адресу 20002 хранится смещение для определения адреса операнда источника (значения переменной VR1), а по адресу 20004 - смещение для определения адреса операнда приемника (значения переменной S).

Для определения адреса операнда источника процессор складывает смещение, хранящееся по адресу 20002, с содержимым регистра PC. После пересылки указанного значения смещения в процессор содержимое регистра PC равно 20004 - поэтому искомое значение адреса операнда источника равно:

смещение + PC = 4 + 20004 = 20010 .

Аналогично определяется адрес операнда приемника:

смещение + PC = 6 + 20006 = 20014 .

23. XOR

Опкод: 074RDD

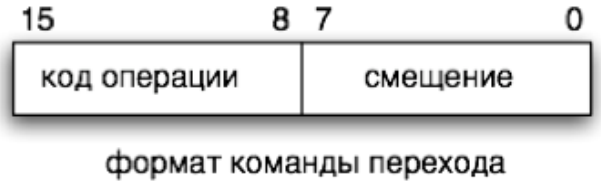
Действие: логическое "исключающее ИЛИ"

Описание:

Логическая операция XOR (исключающее ИЛИ) над регистром R и приемником DD.

Команды перехода

Каждая команда перехода занимает одно слово памяти (кроме команды JMP) и имеет формат:



формат команды перехода

Команды JMP и SOB имеют другой формат.

Команда перехода (ветвления) позволяет изменить содержимое счетчика команд (регистра PC) на величину смещения, указанного в младшем байте кода команды. А изменение содержимого счетчика команд на величину смещения приведет к продолжению выполнения программы с адреса, равного выражению:

АП = АК + 2 + 2 * СМ

где

АП - адрес, с которого продолжится выполнение программы после исполнения команды перехода (адрес перехода);
АК - адрес, где хранится сама команда перехода;
СМ - смещение, указанное в коде команды перехода.

Старший разряд смещения (разряд 7 в коде команды) является знаковым и равен "1" (для отрицательных смещений) или "0" (для положительных смещений). Так как для хранения значения смещения отводится только младший байт кода команды, то оно не может выйти за границы интервала от -128Д до +127Д.

24. BR

Опкод: 0004XXX

Действие: переход.

Описание:

Команда BR осуществляет безусловный переход на определенный адрес памяти (адрес перехода), начиная с которого будет продолжено выполнение программы.

Полный код команды получается сложением кода команды и значения смещения, которое равно выражению

СМ = (АП - АК - 2) / 2 .

Пример:

```
7000: 000401       BR   TT
7002: 005001       CLR  R1
7004: 010102  TT:  MOV  R1,R2
```

Описание примера:

Первая команда производит переход на адрес 7004. Значение смещения определилось так:

СМ = (АП - АК - 2) / 2 = (7004 - 7000 - 2) / 2 = 1 .

Во фрагменте программы на языке ассемблера использована метка TT, обозначающая адрес памяти, где размещена команда MOV R1,R2. Команда BR TT передает управление на адрес, обозначенный ("помеченный") меткой TT.

Пример с отрицательным смещением:

```
35004:  005001  V0:  CLR R1
...
35036:  000762       BR  V0
```

Описание примера:

Команда 000762 производит переход на адрес 35004. Значение смещения определяется так:

СМ = (АП - АК - 2) / 2 = (35004 - 35036 - 2) / 2 = -34 / 2.

Значение смещения - отрицательное число, поэтому оно представлено в дополнительном коде. Полный код команды равен 000762 получается как сумма кода команды 000400 и смещения 362.

25. BNE

Опкод: 0010XXX

Действие: Branch if Not Equal - переход, если не равно.

Описание:

По команде BNE переход произойдет, если разряд Z PSW сброшен в "0".

Пример:

```
1000:  020104        CMP R1,R4
1002:  001001        BNE MET
1004:  010102        MOV R1,R2
1006:  010103 MET:   MOV R1,R3
```

Описание примера:

В этом примере первая команда сравнивает содержимое регистра R1 с содержимым регистра R4. В результате такого сравнения разряд Z PSW установится в "1" (если содержимые регистров R1 и R4 равны) или сбросится в "0" (если содержимые регистров не равны). Если содержимые регистров не равны (если разряд Z PSW сброшен в "0"), то вторая команда 001001 передает управление на адрес, где хранится 4-ая команда 010103, минуя 3-ю команду 010102. В противном случае (если разряд Z PSW установлен в "1") естественный порядок выполнения программы не нарушается - все 4 команды выполняются одна за другой.

26. BEQ

Опкод: 0014XXX

Действие: Branch if Equal - переход, если равно.

Описание:

Команда BEQ является обратной по отношению к команде BNE. Переход по этой команде произойдет, если разряд Z PSW установлен в "1".

27. BPL

Опкод: 1000XXX

Действие: Branch if Plus - переход, если плюс.

Описание:

По команде BPL произойдет переход, если разряд N PSW сброшен в "0".

Пример:

```
1000:  005710        TST (R0)
1002:  100002        BPL PL
1004:  005210        INC (R0)
1006:  004001        BR  MIN
1010:  005310  PL:   DEC (R0)
1012:  011002  MIN:  MOV (R0),R2
```

Описание примера:

Команда 005710 (TST) тестирует (проверяет) содержимое слова памяти, адрес которого хранится в R0. По команде 100002 произойдет переход, если при выполнении предыдущей команды (TST) разряд N PSW был сброшен в "0" (это произойдет, если тестируемый операнд больше или равен 0).

28. BMI

Опкод: 1004XXX

Действие: Branch if Minus - переход, если минус.

Описание:

Команда является обратной по отношению к команде BPL. По команде BMI переход произойдет, если к моменту выполнения команды BMI разряд N PSW установлен в "1".

29. BVC

Опкод: 1020XXX

Действие: Branch if V is Clear - переход, если разряд V PSW очищен (сброшен в "0").

30. BVS

Опкод: 1024XXX

Действие: Branch if V is Set - переход, если разряд V PSW установлен в "1".

31. BGE

Опкод: 0020XX

Действие: Branch if Greater or Equal - переход, если больше или равно.

Описание:

Эту команду можно использовать для перехода после сравнения двух операндов, как чисел со знаком.

Пример:

```
5000:  010103        MOV  R1,R3
5002:  020102        CMP  R1,R2
5004:  002001        BGE  M1
5006:  010203        MOV  R2,R3
5010:   ...    M1:   ...
```

Описание примера:

Команда 020102 (CMP) сравнивает содержимое регистров R1 и R2. По команде 002001 (BGE) произойдет переход, если содержимое регистра R1 оказалось больше или равно содержимому R2. Нетрудно догадаться, что в результате выполнения данной программы в R3 будет находиться максимальное значение из содержащихся в R1 и R2.

32. BLT

Опкод: 0024XXX

Действие: Branch if Less Then - переход, если меньше чем.

Описание:

Если в предыдущем примере вместо команды BGE поставить команду BLT, то переход по команде BLT произойдет, если содержимое R1 окажется меньше содержимого R2, и в результате программа получит в R3 минимальное значение. Подразумевается, что в операции сравнения оба операнда являются числами со знаком.

33. BGT

Опкод: 0030XXX

Действие: Branch if Greater Than - переход, если больше чем.

Описание:

Если команда BGT следует за командой сравнения двух операндов, то переход произойдет, если операнд источника больше операнда приемника. Подразумевается, что в операции сравнения оба операнда являются числами со знаком.

34. BLE

Опкод: 0034XXX

Действие: Branch if Less or Equal - переход, если меньше или равно.

Описание:

Если команда BLE следует за командой сравнения двух операндов, то переход произойдет, если операнд источника меньше или равен операнду приемника. Подразумевается, что в операции сравнения оба операнда являются числами со знаком.

35. BHI

Опкод: 1010XXX

Действие: Branch if Higher - переход, если выше.

Описание:

Команда BHI аналогична команде BGT, но в этом случае операнды рассматриваются не как числа со знаком. Значения 16-разрядных целые чисел без знака находятся в диапазоне от 0 до 65535. Таким способом имеет смысл сравнивать адреса.

36. BLOS

Опкод: 1014XXX

Действие: Branch if Lower or Same - переход, если ниже или столько же.

Описание:

Команда BLOS аналогична команде BLE, но в этом случае операнды рассматриваются не как числа со знаком. Значения 16-разрядных целые чисел без знака находятся в диапазоне от 0 до 65535. Таким способом имеет смысл сравнивать адреса.

37. BHIS(BCC)

Опкод: 1030XXX

Действие: Branch if Higher or Same - переход, если выше или столько же (или бит C PSW равен 0).

Описание:

Команда BHIS аналогична команде BGE, но в этом случае операнды рассматриваются не как числа со знаком. Значения 16-разрядных целые чисел без знака находятся в диапазоне от 0 до 65535. Таким способом имеет смысл сравнивать адреса.

38. BLO(BCS)

Опкод: 1034XXX

Действие: Branch if Lower - переход, если ниже (или бит C PSW равен 1).

Описание:

Команда BLO аналогична команде BLT, но в этом случае операнды рассматриваются не как числа со знаком. Значения 16-разрядных целые чисел без знака находятся в диапазоне от 0 до 65535. Таким способом имеет смысл сравнивать адреса.

39. SOB

Опкод: 077RNN

Действие: Subtract One and Branch, if not equal - вычесть 1 и сделать переход, если не 0.

Описание:

Значение смещения занимает 6 младших разрядов кода команды и рассматривается как число без знака - так как переход по этой команде происходит только в обратном направлении (в сторону уменьшения адресов). Причем переход произойдет только в том случае, если содержимое регистра, указанного в коде команды, после вычитания из него 1 не равно 0.

Для получения кода команды к коду команды 077R00 прибавляется значение смещения, равное значению выражения

(АК + 2 - АП) / 2

где

АК - адрес команды SOB;
АП - адрес перехода.

Значения адресов АК и АП задаются в восьмеричной системе счисления- поэтому все расчеты в указанном выражении ведутся по восьмеричной системе счисления. Например,
(1036 + 2 - 1034) / 2 = 2; (1056 + 2 - 1002) / 2 = 27.

Расчеты значения смещения по данной формуле можно производить, используя десятичные значения адресов АК и АП - но результат после этого должен быть представлен в восьмеричной

системе счисления. Например, код команды без смещения равен 077300 и восьмеричное значение смещения равно 27, тогда код команды, содержащий необходимое значение смещения, будет равен 077327.

Пример:

```
1032:  010102            MOV R1,R2
1034:  000240        M2: NOP
1036:  077202            SOB R2,M2
```

Описание примера:

Пустой цикл, организующий задержку. Чем больше значение регистра R1, тем больше время задержки.

40. JMP

Опкод: 0001DD

Действие: Jump - прыгнуть (перейти).

Описание:

Команда JMP выполняет такие же действия, что и команда BR. Но если применение команды BR ограничено диапазоном смещений, то команду JMP можно использовать для перехода (передачи управления) на любой адрес программы. Поле адресации "DD" в коде команды задает не адрес операнда (поскольку операнда как такового в этой команде нет), а адрес, с которого будет продолжено выполнение программы после исполнения команды JMP. Поэтому в команде JMP недопустимо использование прямого регистрового способа адресации, так как передача управления на регистр процессора не имеет смысла.

Пример:

```
000137 007000    JMP @#7000
```

Описание примера:

После выполнения этой команды программа продолжит свою работу с адреса 7000 - управление будет передано на адрес 7000.

Пример 2:

```
5000: 000137 007554        JMP @#MET
 ...
7554: 005001         MET:  CLR R1
```

Пример 3:

```
5000: 000167 002550        JMP MET
 ...
7554: 005001         MET:  CLR R1
```

Описание примеров:

Результаты выполнения приведенных фрагментов программ одинаковы. Но в первом случае используется абсолютная адресация, а во втором - относительная. Если, например, весь участок программы с адреса 5000 по 7554 переместить в другое место ОЗУ, оператор JMP с относительной адресацией будет работать правильно, поскольку относительное смещение метки MET от команды JMP не изменится. А оператор с абсолютной адресацией отправит программу все равно на адрес 7554, и перемещенная программа будет работать неправильно.

Команды для работы с подпрограммами

41. JSR

Опкод: 004RDD

Действие: Jump to Subroutine - перейти на подпрограмму

Описание:

По команде JSR адрес возврата (адрес команды, следующей за командой JSR) запоминается в регистре, номер которого указывается в коде команды 004RDD вместо буквы "R", а содержимое самого регистра процессора до этого запоминается в стеке (в специально отведенном месте ОЗУ). Если в коде команды указан номер регистра R7 (цифра "7"), то адрес возврата запоминается в стеке.

Назначение поля адресации "DD" в коде команды такое же, что и для команды JMP. Рекомендации по выбору режима адресации те же, что и для команды JMP - в программах (или участках программы), которые могут в процессе работы (или при отладке) перемещаться в ОЗУ, следует использовать относительную адресацию.

42. RTS

Опкод: 00020R

Действие: Return from Subroutine - возврат из подпрограммы

Описание:

Команда RTS возвращает управление на адрес возврата, который по команде JSR был запомнен в регистре процессора. Номер этого регистра должен быть указан в коде данной команды 00020R вместо буквы "R". При выполнении команды содержимое указанного регистра загружается в счетчик команд, а сам регистр загружается значением, взятым из стека (обычно это то значение регистра, которое было запомнено при выполнении команды JSR). Если в коде команды указан регистр R7 (цифра "7"), то запомненный по команде JSR адрес возврата загружается в счетчик команд из стека.

Пример 2:

```
4000:  004737      JSR PC,@#7500
4002:  007500
...    ...          ...
7500:  012700      MOV #14,R0
7502:  000014
7504:  104016      EMT 16
7506:  000207      RTS PC
```

Описание примера:

Команда JSR PC,@#7500 передает управление на подпрограмму, начинающуюся с адреса 7500. Возврат из подпрограммы производится командой RTS PC - выполнение основной программы продолжится с адреса 4004.

43. MARK

Опкод: 0064NN

Действие: При возврате из подпрограммы производится очистка стека от параметров, переданных подпрограмме.

Описание:

Комана MARK предназначена для использования в компиляторах языков высокого уровня - для упрощения процедуры очистки стека от параметров при возврате из подпрограмм. Эта команда выполняется только из стека, куда она должна предварительно помещаться перед вызовом подпрограммы.

При выполнении команды значение SP становится равным PC + (2 * NN), затем значение регистра PC становится равным регистру R5 и последним действием значение регистра R5 восстанавливается из стека.

Типовой код, использующий команду MARK:

```
                                                ; ПОДГОТОВИТЕЛЬНЫЕ ОПЕРАЦИИ
                                                ;  Записать в стек:
        MOV             R5, -(SP)               ; текущее значение R5,
        MOV             #101, -(SP)     ; параметры
        MOV             #102, -(SP)     ; для подпрограммы
        MOV             #103, -(SP)     ; SBR,
        MOV             #6403,-(SP)     ; код команды MARK 3.
        MOV             SP, R5          ; Адрес команды MARK занести в R5
                                                ; для обеспечения перехода к ней.
        ;ВЫЗОВ ПОДПРОГРАММЫ
        JSR             PC, SBR
        ;...........       Текст дальнейшей
        ;...........       программы.

        ;ПОДПРОГРАММА SBR
SBR:;...........       Текст
        ;...........       подпрограммы.
        RTS             R5                              ; Выход из SBR на команду MARK.
```

Команда MARK работает только с регистром R5.

Команды прерываний

Источниками возникновения прерываний могут быть как сигналы внешних устройств (аппаратные прерывания), так и некоторые ситуации, возникающие при выполнении программы (программные прерывания).

Каждый источник прерывания имеет свой вектор прерывания. Это адрес ОЗУ, по которому записан адрес программы обработки данного прерывания.

Для того, чтобы установить обработчик прерывания, необходимо записать адрес его первой команды в соответствующий вектор прерывания. Чтобы обеспечить возврат к прерванной программе, в конце программы обработки прерывания должна стоять команда RTI.

Рассмотрим пример возникновения прерывания от внешнего устройства При возникновении запроса на прерывание процессор заканчивает выполнение текущей команды программы и анализирует бит P в РСП. Если он равен нулю, то процессор начинает обработку прерывания.

Первым делом процессор помещает в стек текущее значение PSW, потом содержимое счетчика команд PC (он показывает на следующую команду после только что выполненной). После этого процессор берет из соответствующего вектора прерывания (например, из ячейки 60) адрес программы обработки прерывания и помещает его в PC, а из следующей по порядку ячейки 62 берет новое значение PSW и помещает его в РСП. Если в 62 ячейке было записано число 200, то теперь бит P в РСП будет установлен, и новое прерывание не будет обрабатываться до тех пор, пока не закончится обработка данного прерывания.

Бит P в PSW маскирует только маскируемые прерывания.

44. EMT

Опкод: 104000 ... 104377

Действие: командное прерывание для системных программ

Описание:

Процессор обрабатывает эту команду таким же образом, как если бы некое внешнее устройство выдало запрос на прерывание по вектору 30. Фактически же эта команда придумана специально, как кратчайший способ вызова системных подпрограмм. Дело в том, что команда EMT выполняется процессором одинаково независимо от содержимого младшего байта команды. Программа обработки прерывания (ее называют EMT-диспетчером) использует младший байт команды EMT как номер подпрограммы из своей системной таблицы и вызывает ее. Свой набор EMT-команд характерен для каждой операционной системы

Выгода при использовании EMT-команд такова: команда EMT вместе с номером вызываемой подпрограммы занимает одно слово, а обычный способ вызова подпрограмм комндой

"JSR PC,адрес" требует двух слов памяти.

Пример:

```
104014        EMT  14
```

Пример обработчика командного прерывания:

```
MOV    R0,-(SP)  ; сохраним регистр в стеке
MOV    2(SP),R0  ; получим из стека адрес возврата, который
                                  ; сохранен там командой TRAP
MOV    -(R0),R0  ; поместить в R0 саму команду - виновницу
                 ; прерывания

...              ; в зависимости от значения младшего байта R0
...              ; выполняются соответствующие действия

MOV    (SP)+,R0  ; восстановить R0
RTI              ; выйти из прерывания
```

45. TRAP

Опкод: 104400 ... 104777

Действие: командное прерывание

Описание:

Команда TRAP действует аналогично EMT, но только по вектору 34.

46. BPT

Опкод: 000003

Действие: прерывание для отладки

47. IOT

Опкод: 000004

Действие: прерывание ввода-вывода

48. RTI

Опкод: 000002

Действие: возврат из прерывания

49. RTT

Опкод: 000006

Действие: возврат из отладочного прерывания

Безоперандные команды

50. HALT

Опкод: 000000

Действие: Halt - останов.

Описание: Вызывает прерывание по вектору 4.

51. WAIT

Опкод: 000001

Действие: Wait - ждать

Описание:

По этой команде процессор временно прекращает выполнение программы и переходит в режим ожидания прерывания. После того, как произойдет прерывание от внешнего устройства, процессор обработает прерывание, и выполнение программы продолжится с команды, следующей за командой WAIT.

52. RESET

Опкод: 000005

Действие: Reset - сброс

Описание:

По этой команде все внешние устройства устанавливаются в состояние, которое они имеют после включения питания, после чего процессор возобновляет работу.

53. NOP

Опкод: 000240

Действие: No OPeration - нет операции

Описание:

По этой команде процессор не выполняет никаких действий и переходит к выполнению следующей команды. Эта команда может использоваться при отладке программы в кодах для замены нескольких удаляемых команд, а также для создания временных задержек при работе программы.

54. CLC

Опкод: 000241

Действие: Clear C - очистить C (разряд PSW)

55. CLV

Опкод: 000242

Действие: Clear V - очистить V (разряд PSW)

56. CLZ

Опкод: 000244

Действие: Clear Z - очистить Z (разряд PSW)

57. CLN

Опкод: 000250

Действие: Clear N - очистить N (разряд PSW)

58. CCC

Опкод: 000257

Действие: Clear Condition Code - очистить коды условий

Описание:

Команда CCC одновременно очищает разряды C,V,Z,N PSW.

59. SEC

Опкод: 000261

Действие: Set C - установить C (разряд PSW)

60. SEV

Опкод: 000262

Действие: Set V - установить V (разряд PSW)

61. SEZ

Опкод: 000264

Действие: Set Z - установить Z (разряд PSW)

62. SEN

Опкод: 000270

Действие: Set N - установить N (разряд PSW)

63. SCC

Опкод: 000277

Действие: Set Condition Code - установить коды условий

Описание:

Команда SCC одновременно устанавливает в "1" разряды C,V,Z,N PSW.

Использование стека

Стеком может служить любая свободная область ОЗУ.

Под стеком понимается область ОЗУ, адресация к ячейкам которой осуществляется определенным образом. Стандартный способ работы со стеком, который используют и все команды прерываний и подпрограмм, осуществляется с помощью регистра SP (Stack Pointer - указатель стека).

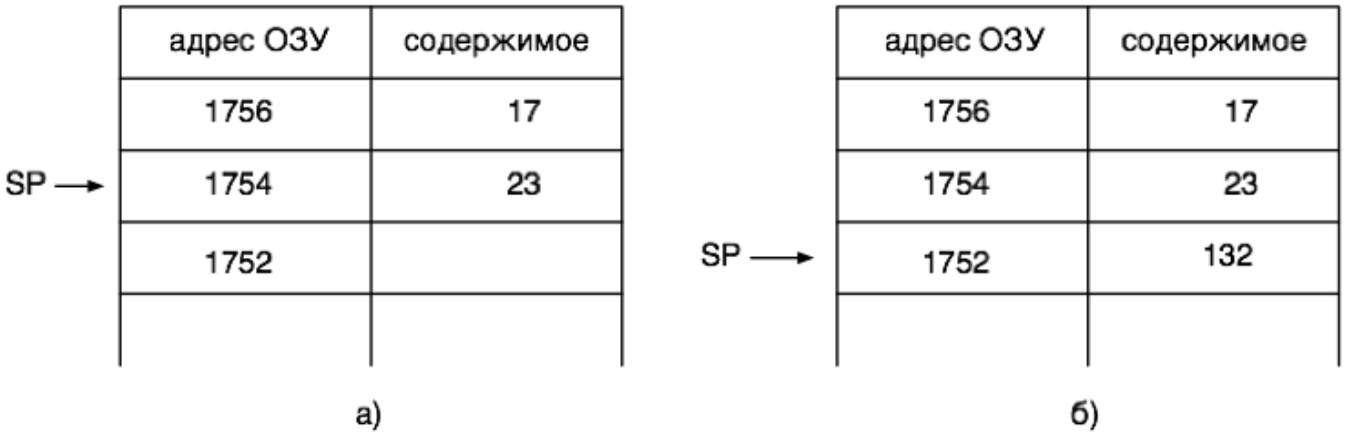
Стек используется программистом для временного хранения промежуточных данных программы. При записи слова в стек используется автодекрементный способ адресации, а при извлечении из стека данных - автоинкрементный.

Процессор использует стек для временного хранения адреса возврата из подпрограммы (или содержимого какого-либо регистра), а также при обработке прерывания. При этом запись в стек и считывание из стека производятся процессором аппаратно.

Перед выполнением команд, использующих стек, в регистр SP процессора необходимо предварительно записать адрес начала стека (настроить стек).

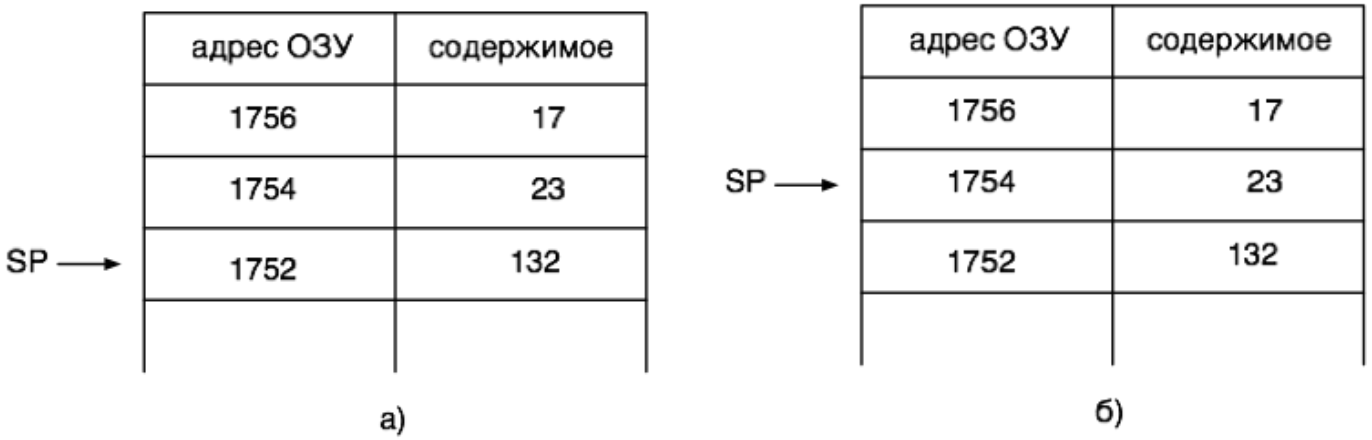
С увеличением размеров стека уменьшается содержимое регистра SP - стек растет в сторону младших адресов.

На следующем рисунке показано выполнение записи в стек по команде "MOV R3,-(SP)". Пусть до выполнения команды MOV регистр SP содержит адрес 1754, а в R3 было записано число 132. Затем указатель стека уменьшается на 2 и содержимое регистра R3 (число 132) записывается по адресу 1752.



Запись в стек: а) до выполнения команды;
б) после выполнения команды.

Дальше на рисунке приведен пример чтения из стека. До выполнения команды "MOV (SP)+,R3" регистр SP содержит адрес 1752. По указанной команде из слова, адрес которого хранится в SP, число 132 пересылается в регистр R3, после чего содержимое регистра SP увеличивается на 2.



Чтение из стека: а) до выполнения команды;
б) после выполнения команды.

Таблица опкодов

N	Код	Мнемоника	Действие	Признаки			
				N	Z	V	C
1	000000	HALT	Останов	-	-	-	-
2	000001	WAIT	Ожидание	-	-	-	-
3	000002	RTI	Возврат из прерывания	*	*	*	*
4	000003	BPT	Командное прерывание для отладки	*	*	*	*
5	000004	IOT	Командное прерывание для ввода-вывода	*	*	*	*
6	000005	RESET	Сброс внешних устройств	-	-	-	-
7	000006	RTT	Возврат из отладочного прерывания	*	*	*	*
8	0001DD	JMP	Безусловный переход	-	-	-	-
9	00020R	RTS	Возврат из подпрограммы	-	-	-	-
10	004RDD	JSR	Обращение к подпрограмме	-	-	-	-

11	104000 ... 104377	EMT	Командное прерывание для системных программ	*	*	*	*
12	104400 ... 104777	TRAP	Командное прерывание	*	*	*	*
13	000240	NOP	Нет операции	-	-	-	-
14	000241	CLC	Очистка C	-	-	-	0
15	000242	CLV	Очистка V	-	-	0	-
16	000244	CLZ	Очистка Z	-	0	-	-
17	000250	CLN	Очистка N	0	-	-	-
18	000261	SEC	Установка C	-	-	-	1
19	000262	SEV	Установка V	-	-	1	-
20	000264	SEZ	Установка Z	-	1	-	-
21	000270	SEN	Установка N	1	-	-	-
22	000277	SCC	Установка всех разрядов (N, Z, V, C)	1	1	1	1
23	000257	CCC	Очистка всех разрядов (N, Z, V, C)	0	0	0	0
24	0003DD	SWAB	Перестановка байтов	*	*	0	0
25	*050DD	CLR(B)	Очистка	0	1	0	0
26	*051DD	COM(B)	Инвертирование	*	*	0	1
27	*052DD	INC(B)	Прибавление единицы	*	*	*	-
28	*053DD	DEC(B)	Вычитание единицы	*	*	*	-
29	*054DD	NEG(B)	Изменение знака	*	*	*	*
30	*055DD	ADC(B)	Прибавление переноса	*	*	*	*
31	*056DD	SBC(B)	Вычитание переноса	*	*	*	*
32	*057DD	TST(B)	Проверка	*	*	0	0
33	*060DD	ROR(B)	Циклический сдвиг вправо	*	*	*	*
34	*061DD	ROL(B)	Циклический сдвиг влево	*	*	*	*
35	*062DD	ASR(B)	Арифметический сдвиг вправо	*	*	*	*
36	*063DD	ASL(B)	Арифметический сдвиг влево	*	*	*	*
37	0064NN	MARK	Восстановление указателя стека	-	-	-	-
38	0067DD	SXT	Расширение знака	-	*	0	-
39	1064SS	MTPS	Запись слова состояния процессора	*	*	*	*
40	1067DD	MFPS	Чтение слова состояния процессора	*	*	0	-
41	*1SSDD	MOV(B)	Пересылка	*	*	0	-
42	*2SSDD	CMP(B)	Сравнение	*	*	*	*
43	*3SSDD	BIT(B)	Проверка разрядов	*	*	0	-
44	*4SSDD	BIC(B)	Очистка разрядов	*	*	0	-
45	*5SSDD	BIS(B)	Логическое сложение	*	*	0	-
46	074RDD	XOR	Исключающее ИЛИ	*	*	0	-
47	06SSDD	ADD	Сложение	*	*	*	*

48	16SSDD	SUB	Вычитание	*	*	*	*
49	0004XXX	BR	Ветвление безусловное	-	-	-	-
50	0010XXX	BNE	Ветвление, если не равно (нулю)	-	-	-	-
51	0014XXX	BEQ	Ветвление, если равно (нулю)	-	-	-	-
52	0020XXX	BGE	Ветвление, если больше и равно (нулю)	-	-	-	-
53	0024XXX	BLT	Ветвление, если меньше (нуля)	-	-	-	-
54	0030XXX	BGT	Ветвление, если больше (нуля)	-	-	-	-
55	0034XXX	BLE	Ветвление, если меньше или равно (нулю)	-	-	-	-
56	077RNN	SOB	Вычитание единицы и ветвление	-	-	-	-
57	1000XXX	BPL	Ветвление, если плюс	-	-	-	-
58	1004XXX	BMI	Ветвление, если минус	-	-	-	-
59	1010XXX	BHI	Ветвление, если больше	-	-	-	-
60	1014XXX	BLOS	Ветвление, если меньше или равно	-	-	-	-
61	1020XXX	BVC	Ветвление, если нет арифметического переполнения	-	-	-	-
62	1024XXX	BVS	Ветвление, если арифметическое переполнение	-	-	-	-
63	1030XXX	BHIS, BCC	Ветвление, если больше или равно	-	-	-	-
64	1034XXX	BLO, BCS	Ветвление, если меньше	-	-	-	-

Обозначения для признаков:

***** - признак принимает значение согласно результату выполнения инструкции;
- - инструкция не влияет на признак;
0 - после выполнения инструкции признак будет равен нулю;
1 - после выполнения инструкции признак будет равен единице.

Источники:

1. Митрюхин В.К., Донской А.Н., Михайлов А.В., Немов А.М. "Программирование на БК 0010-01".
2. Журнал "Микропроцессорные средства и системы" N4, 1984г.
3. Электроника БК 0010, программное обеспечение микро-ЭВМ. Драйвер-мониторная система. Руководство системного программиста.
4. Еремин Е.А. Как работает команда MARK. Информатика и образование, 1991, N6, с.75-76.