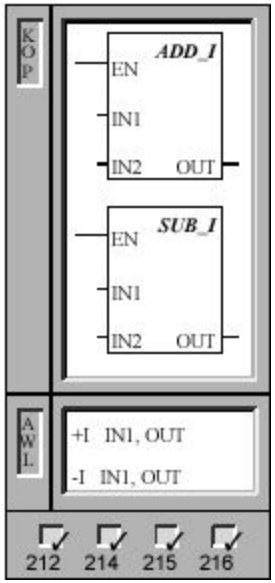
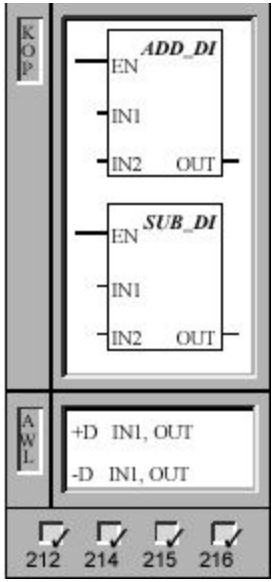
Арифметические операции , инкрементирование и декрементирование

Операции **Сложение целых чисел (16 бит)** и **Вычитание целых чисел (16 бит)** складывают или вычитают два целых числа (16 бит) и передают результат (16 бит) в OUT.



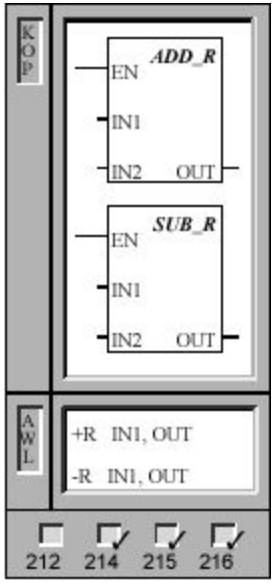
Операнды : IN , IN2: VW, T, Z, EW, AW, MW, SMW, AC, AEW, константа , *VD, *AC, SW
OUT: VW, T, Z, EW, AW, MW, SMW, AC, *VD, *AC, SW

Операции **Сложение целых чисел (32 бита)** и **Вычитание целых чисел (32 бита)** складывают или вычитают два целых числа (32 бита) и передают результат (32 бита) в OUT.



Операнды : IN , IN2: VD, ED, AD, MD, SMD, AC, константа , *VD, *AC, SD
OUT: VD, ED, AD, MD, SMD, AC, *VD, *AC, SD

Операции **Сложение действительных чисел** и **Вычитание действительных чисел** складывают или вычитают два действительных числа (32 бита) и передают действительное число в качестве результата в OUT.

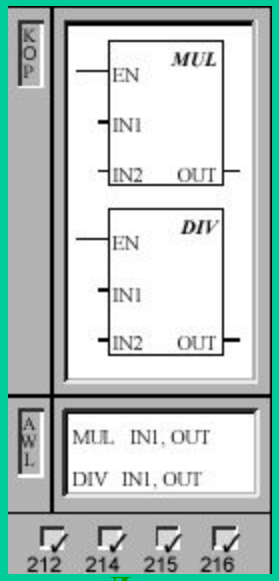


Операнды : IN , IN2: VD, ED, AD, MD, SMD, AC, HC, константа , *VD, *AC, SD
OUT: VD, ED, AD, MD, SMD, AC, *VD, *AC, SD

Указание : При программировании в KOP Вы можете указать , что IN совпадает с OUT. Таким образом , Вы экономите место в памяти .
Эти операции влияют на следующие специальные меркеры :
SM 1 .0 (нуль); SM 1.1 (переполнение); SM 1.2 (отрицательный результат)

Арифметические операции , инкрементирование и декрементирование

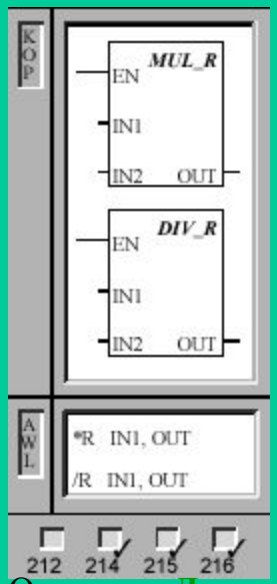
Операция **Умножение целых чисел (16 бит)** умножает два целых числа (16 бит) и передает результат (32 бита) в OUT.



Операнды : IN , IN2: VW, T, Z, EW, AW, MW, SMW, AC, AEW, константа , *VD, *AC, SW
OUT: VD, ED, AD, MD, SMD, AC, *VD, *AC, SW

Операция **Деление целых чисел (16 бит)** делит два целых числа (16 бит) и передает результат (32 бита) в OUT. Результат (32 бита) в OUT состоит из частного (16 младших битов) и остатка от деления (16 старших битов).

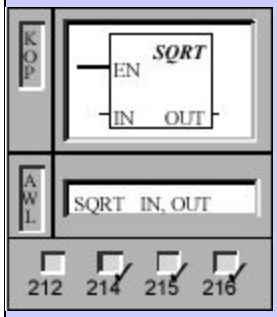
Операция **Умножение действительных чисел** умножает два действительных числа(32 бита) и передает результат (32 бита) в OUT.



Операнды : IN , IN2: VD, ED, AD, MD, SMD, AC, константа , *VD, *AC, SD
OUT: VD, ED, AD, MD, SMD, AC, *VD, *AC, SD

Операция **Деление действительных чисел** делит два действительных числа (32 бита) и передает результат (32 бита) в OUT.

Операция **Извлечение квадратного корня из действительного числа** извлекает квадратный корень из действительного числа (32 бита), заданного в IN. Результат (OUT) тоже является действительным числом (32 бита).

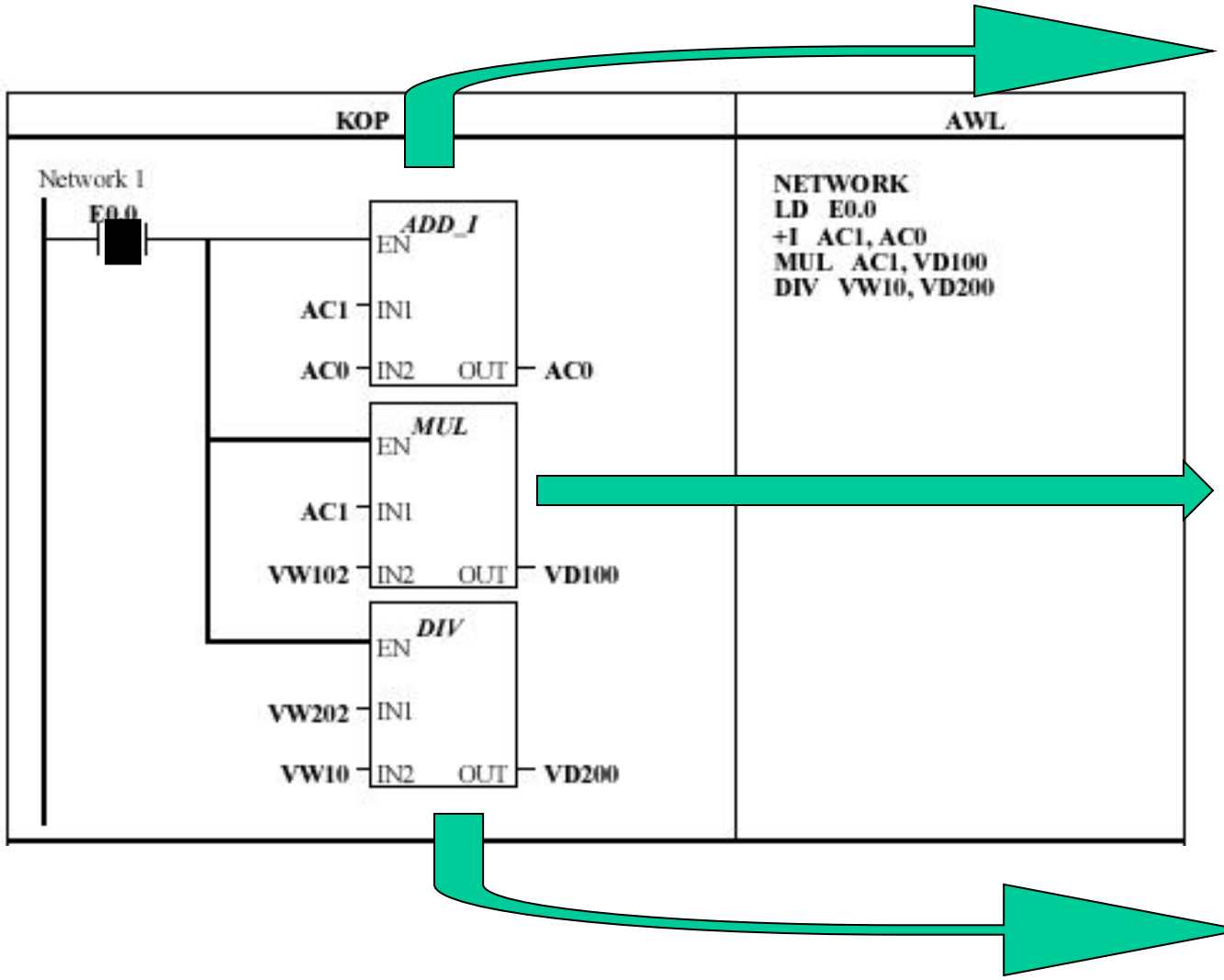


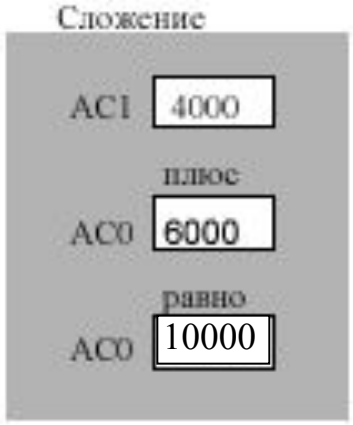
Операнды : IN: VD, ED, AD, MD, SMD, AC, константа , *VD, *AC, SD
OUT: VD, ED, AD, MD, SMD AC, *VD, *AC, SD

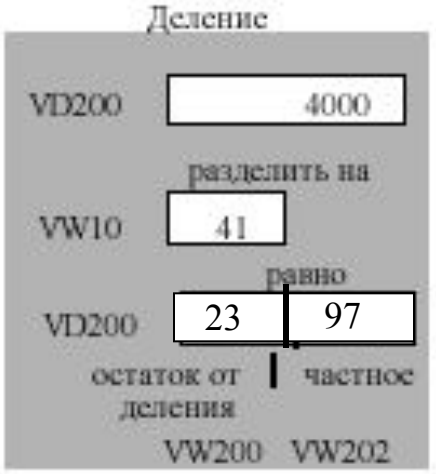
Эта операция влияет на следующие специальные меркеры SM1.0; SM1.1; SM1.2

Эти операции влияют на следующие специальные меркеры :
SM 1 .0 (нуль); SM 1.1 (переполнение); SM 1.2 (отрицательный результат);SM 1.3 (деление на нуль)

Примеры арифметических операций

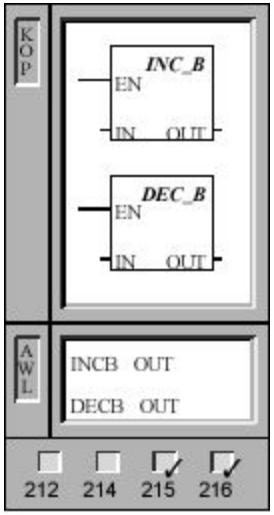






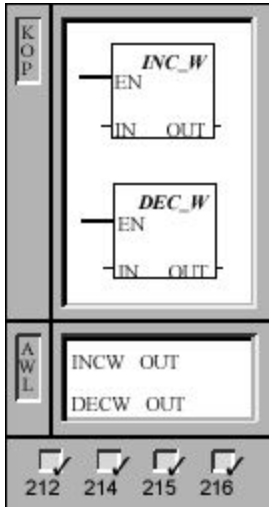
Арифметические операции , инкрементирование и декрементирование

Операции **Увеличить байт на 1** и **Уменьшить байт на 1** прибавляет или вычитает “1” из значения входного байта . Операции не учитывают знака .



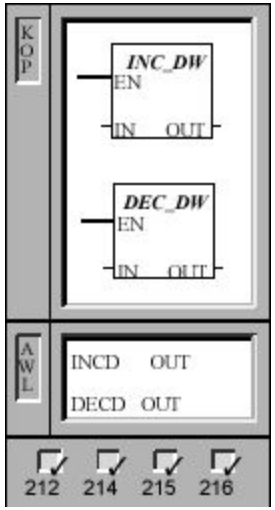
Операнды : IN: VB, EB, AB, MB, SMB, SB, AC, константа , *VD, *AC, SB
OUT: VB, EB, AB, MB, SMB, SB, AC, *VD, *AC, SB

Операции **Увеличить слово на 1** и **Уменьшить слово на 1** прибавляет или вычитает “1” из значения входного слова . Операции учитывают знак (16#7FFF > 16#8000).



Операнды : IN: VW, T, Z, EW, AW, MW, SMW, AC, AEW, константа , *VD, *AC, SW
OUT: VW, T, Z, EW, AW, MW, SMW, AC, *VD, *AC, SW

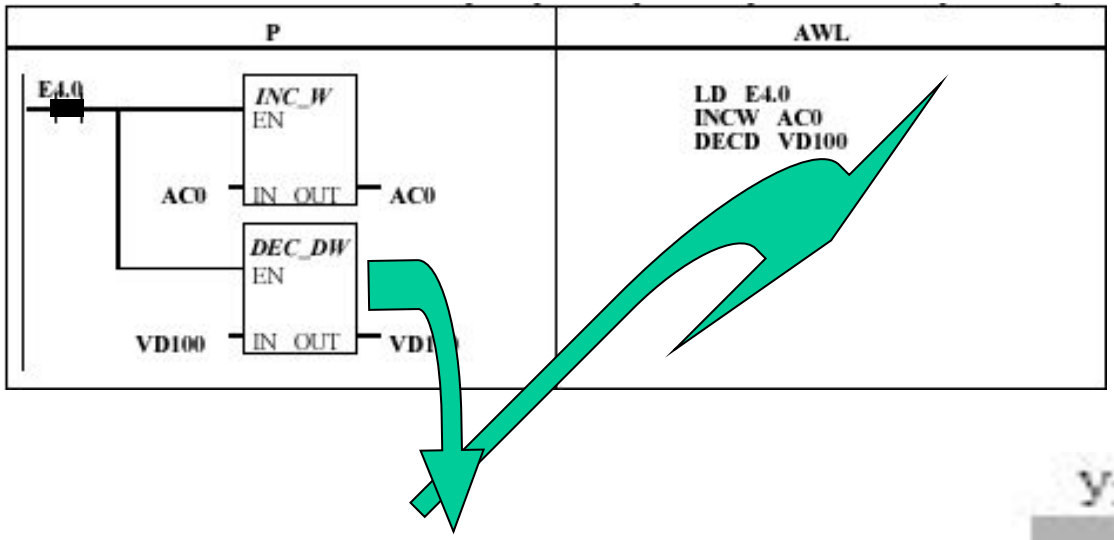
Операции **Увеличить двойное слово на 1** и **Уменьшить двойное слово на 1** прибавляет или вычитает “1” из значения входного двойного слова . Операции учитывают знак (16#7FFFFFFF > 16#80000000).

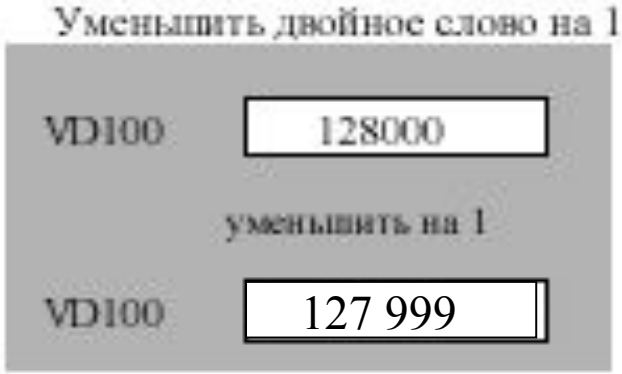


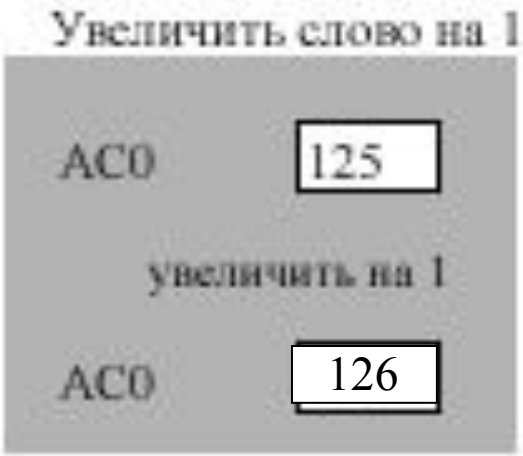
Операнды :IN: VD, ED, AD, MD, SMD, AC, HC, константа , *VD, *AC, SD
OUT: VD, ED, AD, MD, SMD, AC, *VD, *AC, SD

Указание : При программировании в KOP Вы можете указать , что IN совпадает с OUT. Таким образом , Вы экономите место в памяти .
Эти операции влияют на следующие специальные меркеры :
SM 1 .0 (нуль); SM 1.1 (переполнение); SM 1.2 (отрицательный результат)

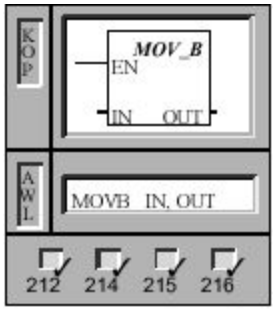
Примеры инкрементирования и декрементирования





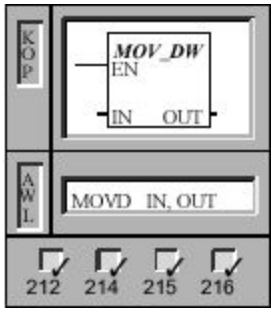


Операции перемещения



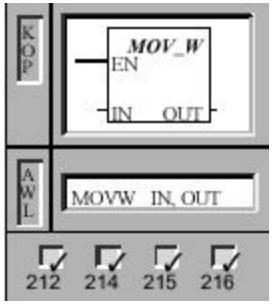
Операция **Передача байта** передает входной байт (IN) в выходной байт (OUT). Входной байт при этом не изменяется .

Операнды : IN: VB, EB, AB, MB, SMB, AC, константа , *VD, *AC, SB
OUT: VB, EB, AB, MB, SMB, AC, *VD*AC, SB



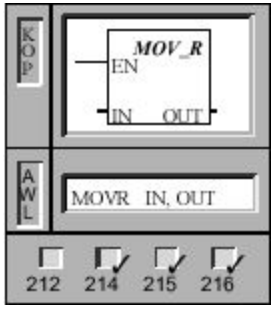
Операция **Передача двойного слова** передает входное двойное слово (IN) в выходное двойное слово (OUT). Входное двойное слово при этом не изменяется .

Операнды : IN: VD, ED, AD, MD, SMD, AC, HC, константа , *VD, *AC, &VB, &EB,&AB, &MB, &T, &Z, &SB, SD
OUT: VD, ED, AD, MD, SMD, AC, *VD, *AC, SD



Операция **Передача слова** передает входное слово (IN) в выходное слово (OUT). Входное слово при этом не изменяется .

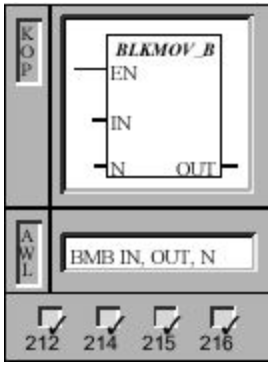
Операнды : IN: VW, T, Z, EW, AW, MW, SMW, AC, AEW, константа *VD, *AC, SW
OUT: VW, T, Z, EW, AW, MW, SMW, AC, AAW, *VD, *AC, SW



Операция **Передача действительного числа** передает входное действительное число (слово 32 бита) (IN) в выходное двойное слово (OUT). Входное двойное слово при этом не изменяется .

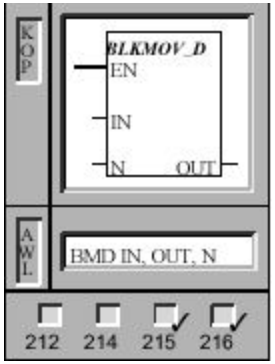
Операнды : IN: VD, ED, AD, MD, SMD, AC, HC, константа , *VD, *AC,SD
OUT: VD, ED, AD, MD, SMD, AC, *VD, *AC, SD

Операции перемещения



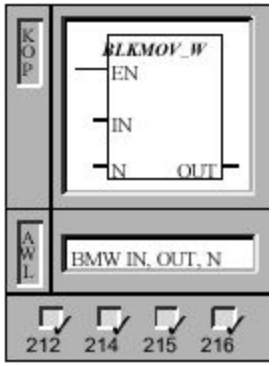
Операция **Блочная передача байтов** передает заданное количество байтов (N) из входного массива , начинающегося с IN, в выходной массив , начинающейся с OUT. N может лежать в диапазоне от 1 до 255.

Операнды : IN, OUT: VB, EB, AB, MB, SMB, *VD, *AC, SB
N: VB, EB, AB, MB, SMB, AC, константа , *VD, *AC, SB

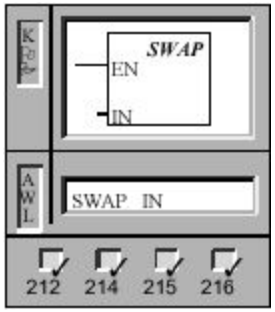


Операция **Блочная передача двойных слов** передает заданное количество двойных слов (N) из входного массива , начинающегося с IN, в выходной массив , начинающийся с OUT.
N может лежать в диапазоне от 1 до 255.

Операнды : IN, OUT: VD, ED, AD, MD, SMD, *VD, *AC, SD
N: VB, EB, AB, MB, SMB, AC, константа , *VD, *AC, SB



Операция **Блочная передача слов** передает заданное количество слов (N) из входного массива , начинающегося с IN, в выходной массив , начинающийся с OUT.
N может лежать в диапазоне от 1 до 255.

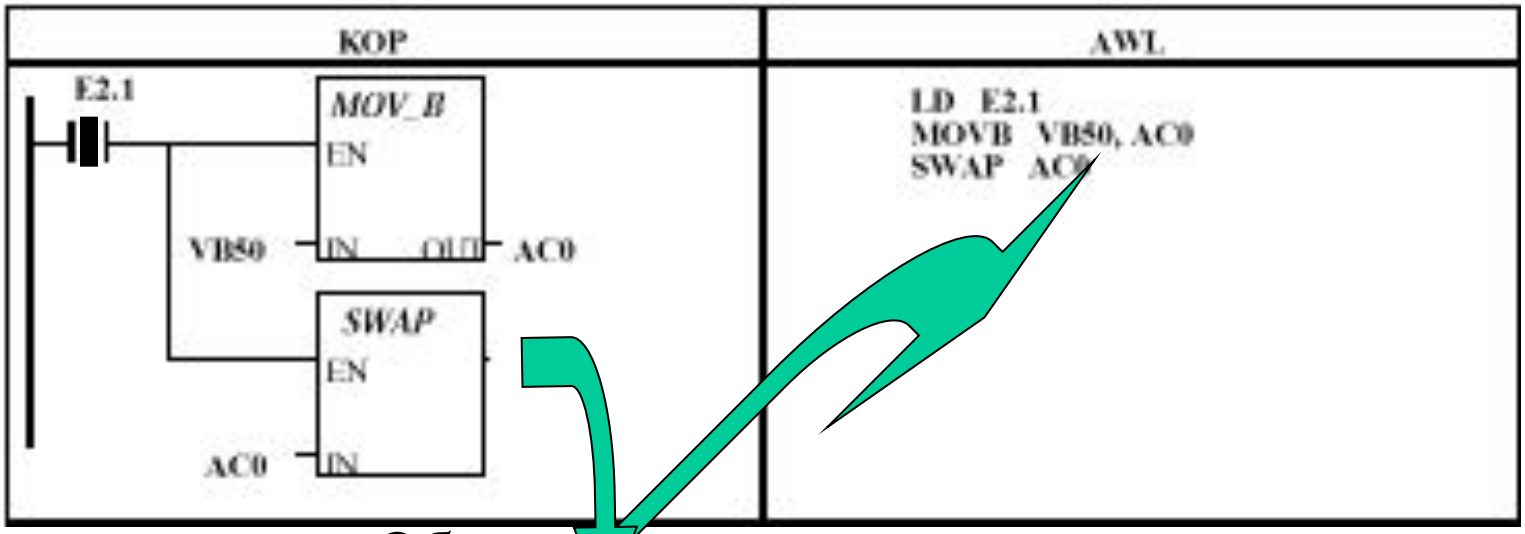


Операция **Обмен байтов в слове** меняет местами старший и младший
байты в слове (IN)..

Операнды : IN, OUT: VW, T, Z, EW, AW, MW, SMW, AEW,*VD, *AC, SW
N: VB, EB, AB, MB, SMB, AC, константа , *VD, *AC, SB

Операнды :IN: VW, T, Z, EW, AW, MW, SMW, SW,AC, *VD, *AC, SW

Пример операций перемещений



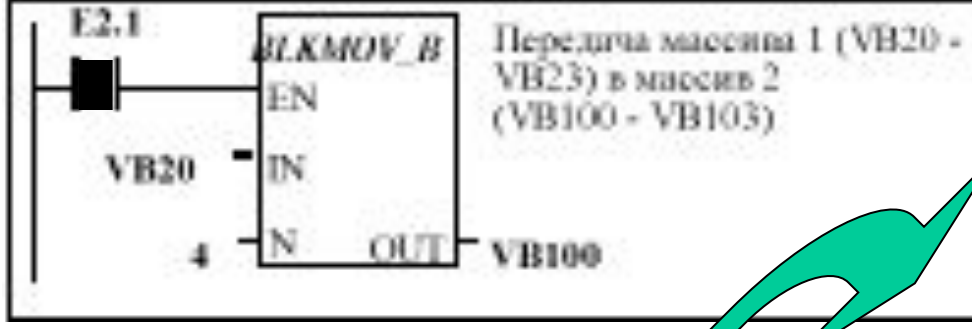
Обмен

AC 0	D6 C 3
AC 0	C3 D6

Передача

VB50	C 3
AC 0	C 3

Пример операции блочной передачи

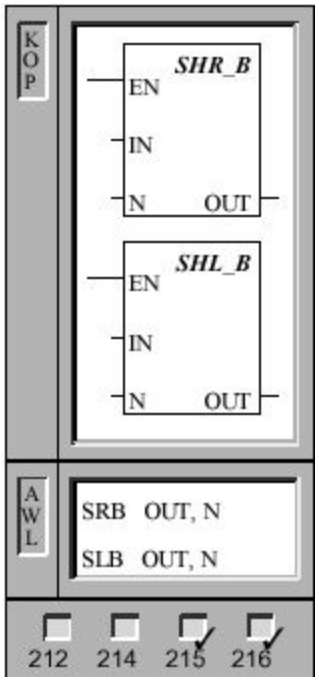
КОР	AWL
	LD E2.1 BMB VB20, VB100, 4

	VB 20	VB 21	VB 22	VB 23
Массив 1	30	31	32	33

	VB 100	VB 101	VB 102	VB 103
Массив 2	30	31	32	33

Операции сдвига и циклического сдвига

Операции **Сдвиг байта вправо** и **Сдвиг байта влево** сдвигают значение байта (IN) на величину сдвига (N) вправо или влево и загружают результат в выходной байт (OUT).

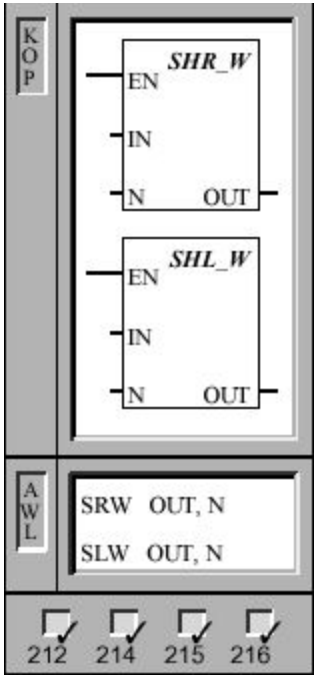


Операнды :IN:
VB, EB, AB, MB, SMB, SB, AC, *VD, *AC

N: VB, EB, AB, MB, SMB, SB, AC,константа , *VD, *AC

OUT: VB, EB, AB, MB, SMB, SB, AC, *VD,*AC

Операции **Сдвиг слова вправо** и **Сдвиг слова влево** сдвигают значение слова (IN) на величину сдвига (N) вправо или влево и загружают результат в выходное слово (OUT).

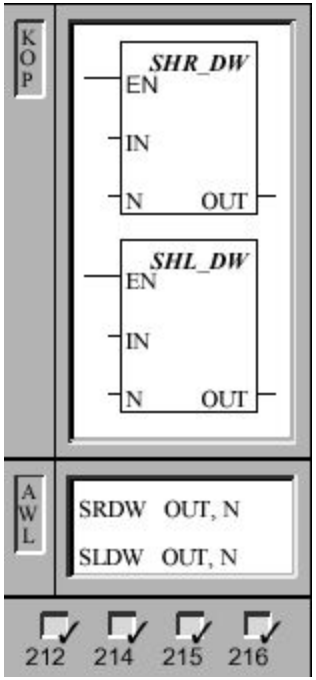


Операнды :IN:
VW, T, Z, EW, MW, SMW, AC, AW,AEW, константа ,*VD, *AC, SW

N: VB, EB, AB, MB, SMB, AC, константа , *VD, *AC, SB

OUT: VW, T, Z, EW, AW, MW, SMW, AC,*VD, *AC, SW

Операции **Сдвиг двойного слова вправо** и **Сдвиг двойного слова влево** сдвигают значение двойного слова (IN) на величину сдвига (N) вправо или влево и загружают результат в выходное двойное слово (OUT).



Операнды :IN:
VD, ED, AD, MD, SMD, AC, HC, константа , *VD, *AC, SD

N: VB, EB, AB, MB, SMB, AC, константа , *VD, *AC, SB

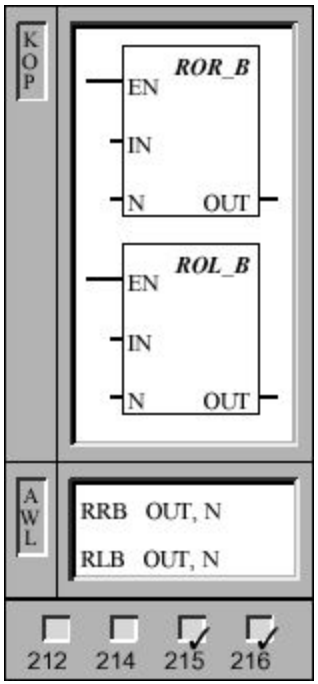
OUT: VD, ED, AD, MD, SMD, AC, *VD, *AC, SD

* Эти операции не учитывают знака
* Эти операции заполняют места «выдвигаемых» битов нулями
* Эти операции влияют на меркеры:

SM 1.0(нуль), SM 1.1(переполнение), который принимает значение «выдвигаемого» бита

Операции сдвига и циклического сдвига

Операции **Циклический сдвиг байта вправо** и **Циклический сдвиг байта влево** циклически сдвигают значение байта (IN) на величину сдвига (N) вправо или влево и загружают результат в выходной байт (OUT).

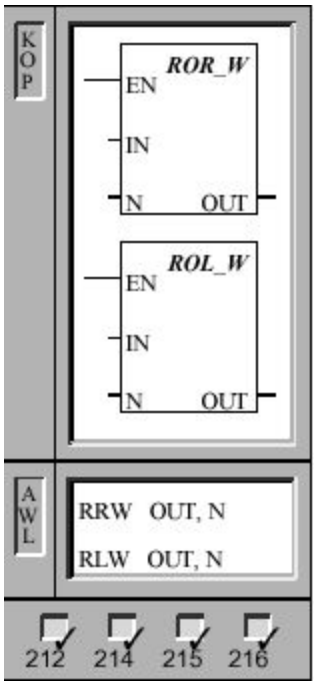


Операнды :IN: VB, EB, AB, MB, SMB, SB, AC, *VD, *AC, SB

N: VB, EB, AB, MB, SMB, SB, AC,константа , *VD, *AC, SB

OUT: VB, EB, AB, MB, SMB, SB, AC, *VD, *AC, SB

Операции **Циклический сдвиг слова вправо** и **Циклический сдвиг слова влево** циклически сдвигают значение слова (IN) на величину сдвига (N) вправо или влево и загружают результат в выходное слово (OUT).

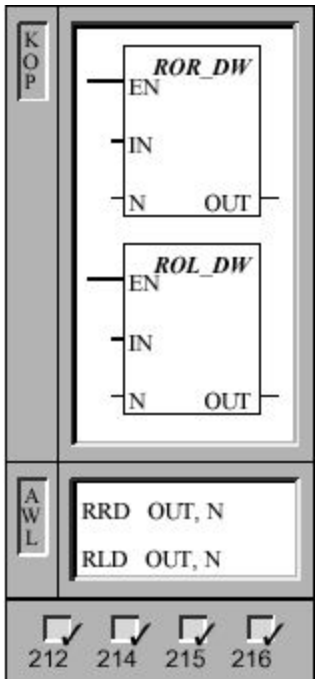


Операнды :IN: VW, T, Z, EW, MW, SMW, AC, AW, AEW, константа , *VD, *AC, SW

N: VB, EB, AB, MB, SMB, AC, константа , *VD, *AC, SB

OUT: VW, T, Z, EW, AW, MW, SMW, AC, *VD, *AC, SW

Операции **Циклический сдвиг двойного слова вправо** и **Циклический сдвиг двойного слова влево** циклически сдвигают значение двойного слова (IN) на величину сдвига (N) вправо или влево и загружают результат в выходное двойное слово (OUT).



Операнды :IN: VD, ED, AD, MD, SMD, AC, HC, константа , *VD, *AC, SD

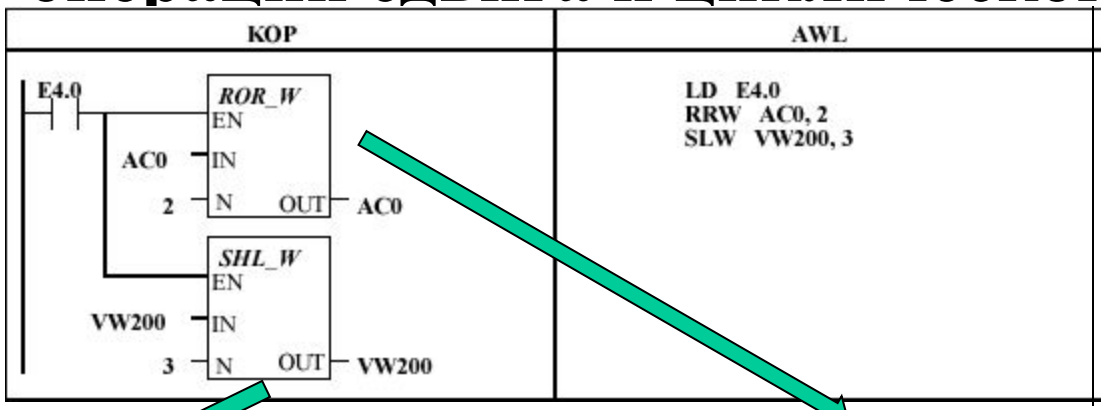
N: VB, EB, AB, MB, SMB, AC, константа , *VD, *AC, SB

OUT: VD, ED, AD, MD, SMD, AC, *VD, *AC, SD

* Эти операции не учитывают знака
* Перед сдвигом выполняется операция по соответствующему модулю (для Байта= 8 и т.д)
* Эти операции влияют на меркеры:

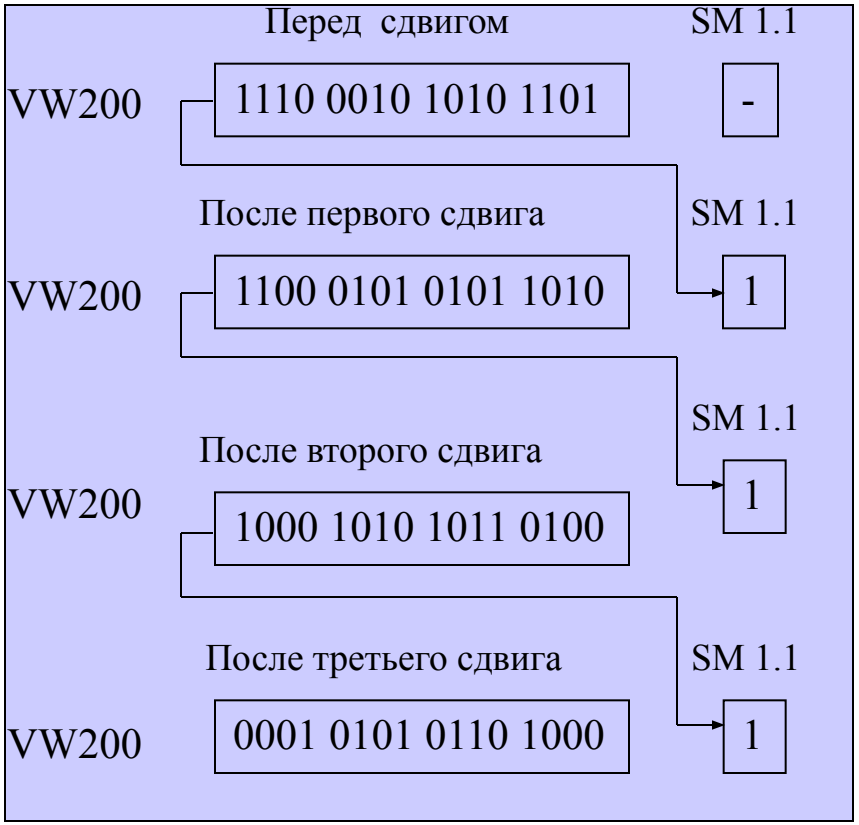
SM 1.0(нуль), SM 1.1(переполнение), который принимает значение «выдвигаемого» бита

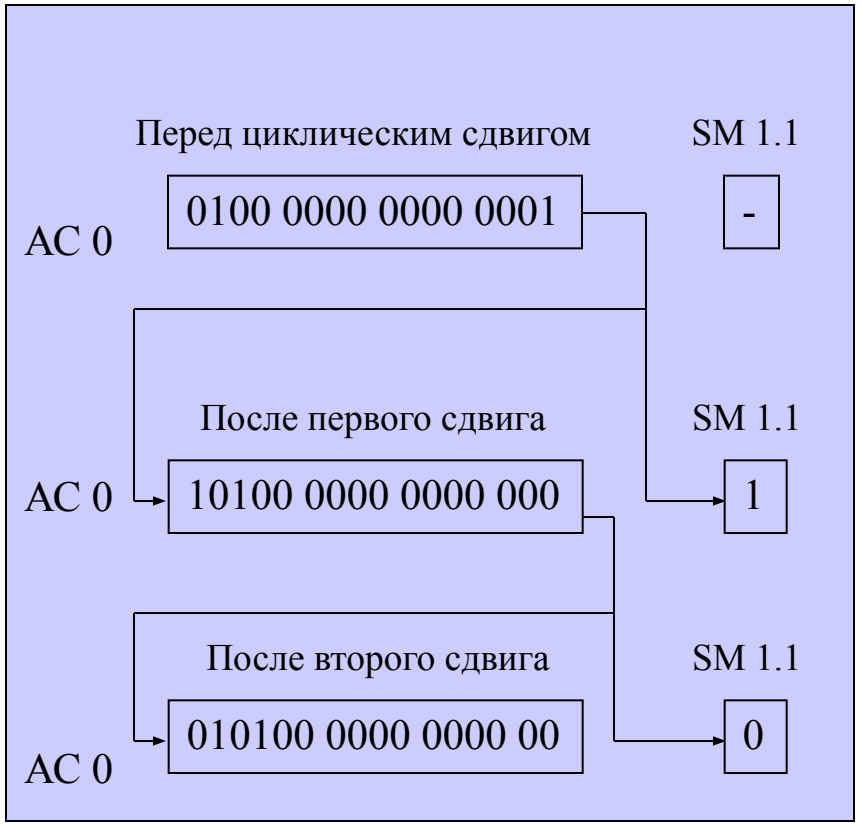
Пример операции сдвига и циклического сдвига



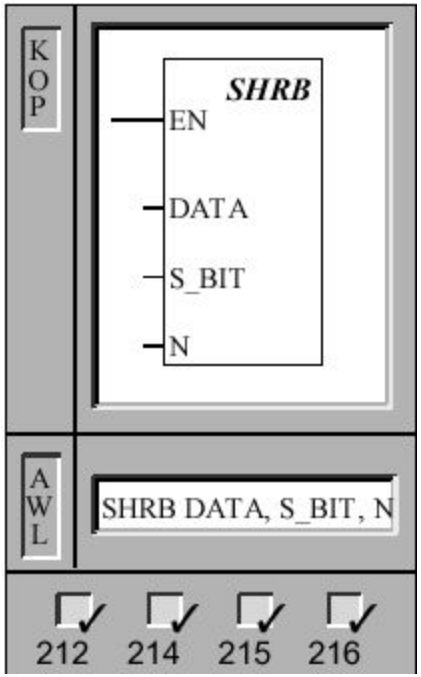
Сдвиг слова влево

Циклический сдвиг слова вправо



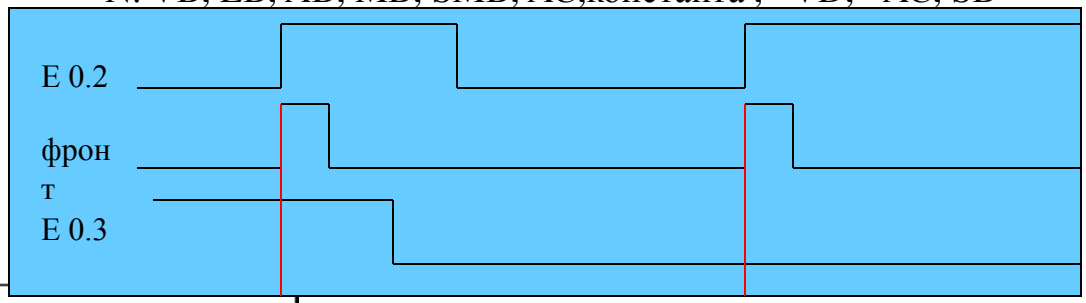


Сдвиг бита в регистре



Операция **Ввод значения в регистр сдвига** вводит значение DATA в регистр сдвига . S_BIT задает младший бит регистра сдвига . N показывает длину регистра сдвига и направление , в котором происходит сдвиг (сдвиг = N, отрицательный сдвиг = –N).

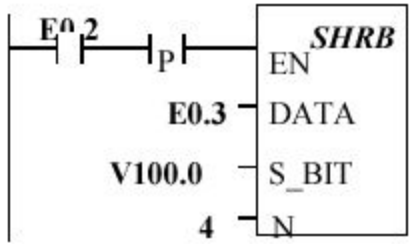
Операнды : DATA, S_BIT: E, A, M, SM, T, Z, V, S
N: VB, EB, AB, MB, SMB, AC,константа , *VD, *AC, SB



Первая операция сдвига

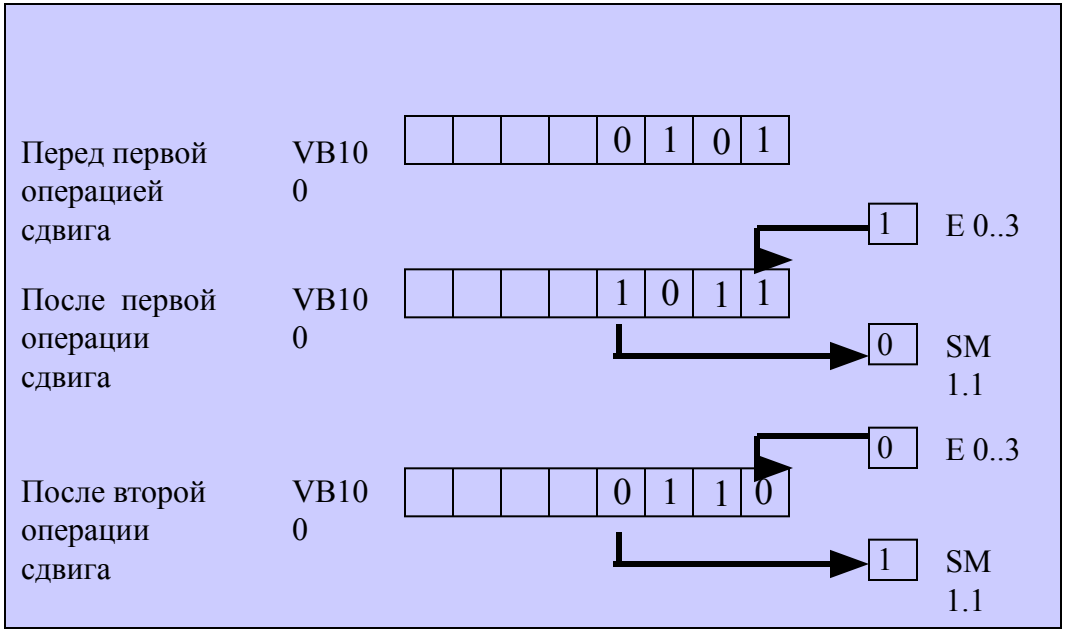
Вторая операция сдвига

KOP

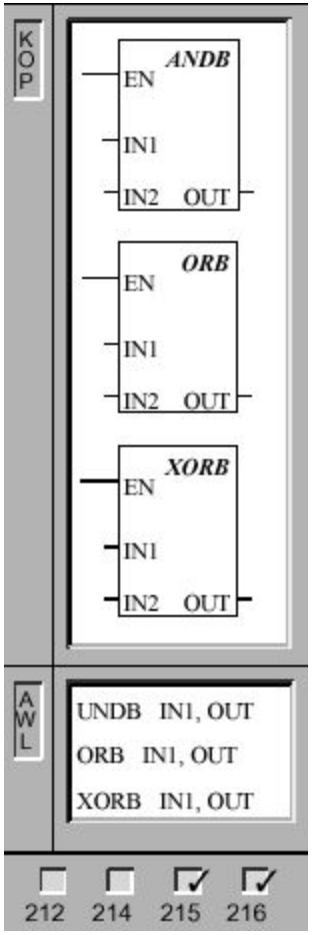


AWL

```
LD   E0.2
EU
SHRB   E0.3, V100.0, 4
```



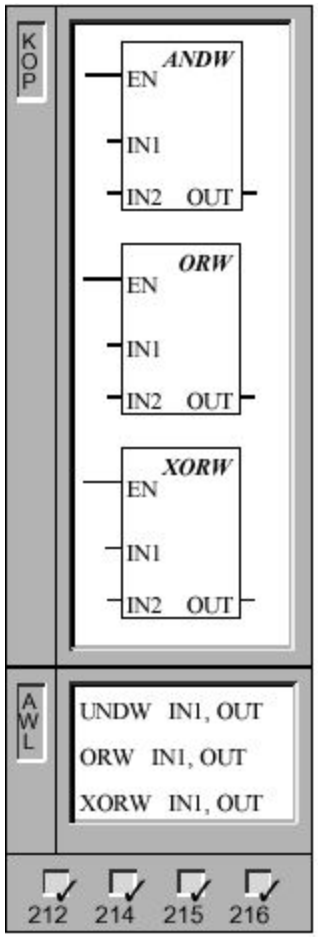
Логические операции



Логическое сопряжение байтов через «И»

Логическое сопряжение байтов через «ИЛИ»

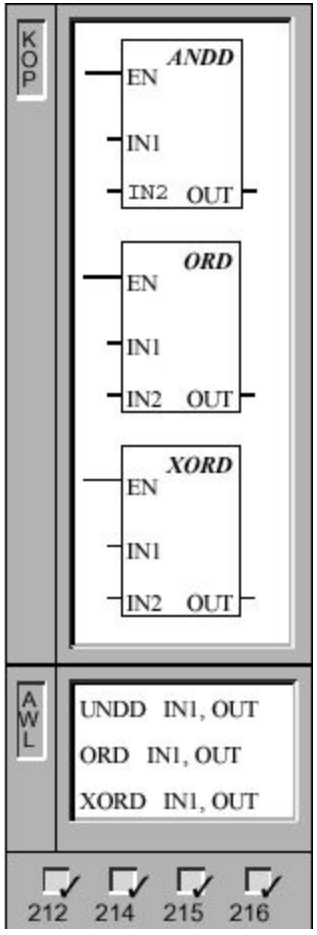
Логическое сопряжение байтов через «ИСКЛЮЧАЮЩЕЕ ИЛИ»



Логическое сопряжение слов через «И»

Логическое сопряжение слов через «ИЛИ»

Логическое сопряжение слов через «ИСКЛЮЧАЮЩЕЕ ИЛИ»



Логическое сопряжение дв. слов через «И»

Логическое сопряжение дв. слов через «ИЛИ»

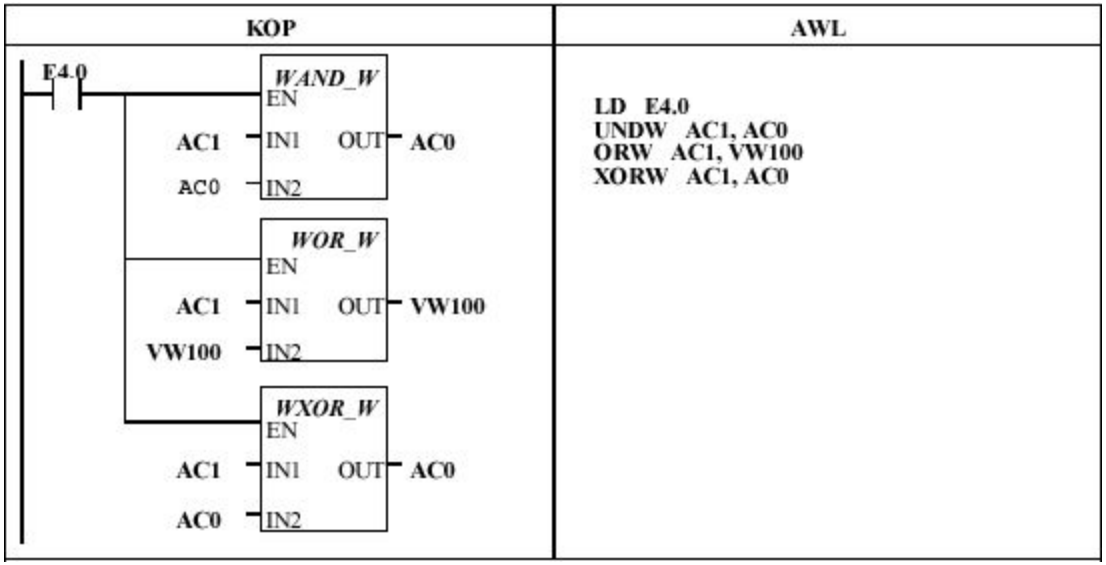
Логическое сопряжение дв. слов через «ИСКЛЮЧАЮЩЕЕ ИЛИ»

Операнды :
IN , IN2: VB, EB, AB, MB, SMB,SB, AC,константа ,*VD, *AC,SB
OUT: VB, EB, AB, MB, SMB, SB, AC,*VD, *AC, SB

Операнды :
IN , IN2:VW, T, Z, EW, AW, MW, SMW, AC,AEW, константа , *VD, *AC, SW
OUT: VW, T, Z, EW, AW, MW, SMW, AC,*VD, *AC, SW

Операнды :
:IN , IN2: VD, ED, AD, MD, SMD, AC, HC,константа , *VD, *AC, SD
OUT: VD, ED, AD, MD, SMD, AC, *VD,*AC, SD

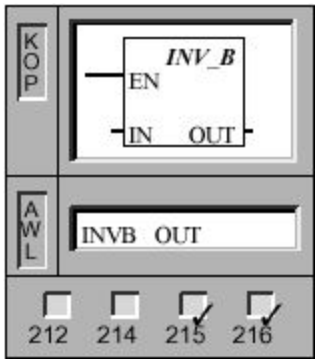
Примеры логических операций



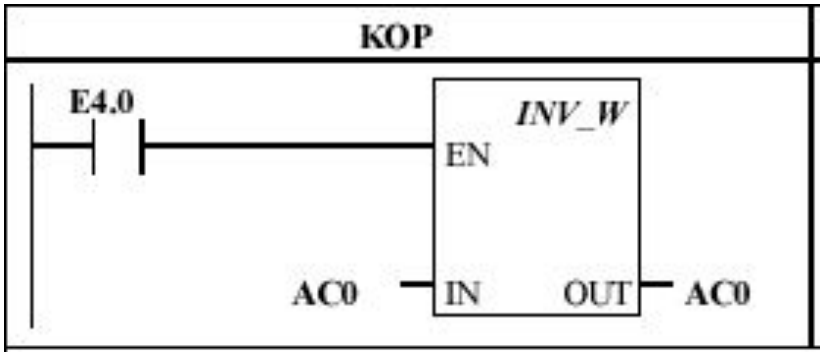
И		ИЛИ		ИСКЛЮЧАЮЩЕЕ ИЛИ	
AC1	0001 1111 0110 1101	AC1	0001 1111 0110 1101	AC1	0001 1111 0110 1101
И		ИЛИ		ИСКЛЮЧАЮЩЕЕ ИЛИ	
AC0	1101 0011 1110 0110	VW100	1101 0011 1010 0000	AC0	0001 0011 0110 0100
равно		равно		равно	
AC0	0001 0011 0110 0100	VW100	1101 1111 1110 1101	AC0	0000 1100 0000 1001

Инверсия данных

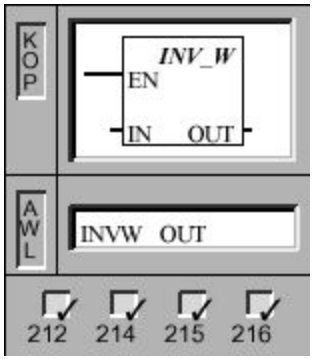
Операция **Образование дополнения до единицы для байта**



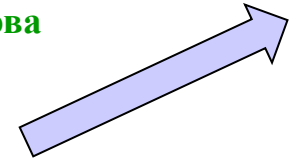
Операнды :IN:VB, EB, AB, MB, SMB, SB, AC, *VD, *AC, SB
OUT:VB, EB, AB, MB, SMB, SB, AC,*VD, *AC, SB

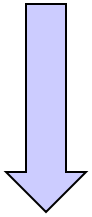


Операция **Образование дополнения до единицы для слова**



Операнды : IN: VW, T, Z, EW, AW, MW, SMW, AC,AEW, константа , *VD, *AC, SW
OUT: VW, T, Z, EW, AW, MW, SMW, AC,*VD, *AC, SW





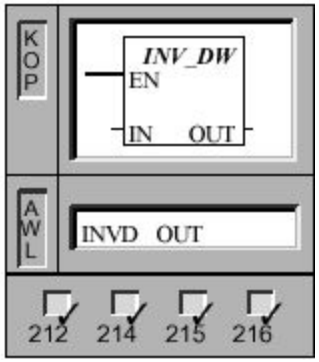
Обращение слова

AC0 1101 0111 1001 0101

Образование дополнения до 1

AC0 0010 1000 0110 1010

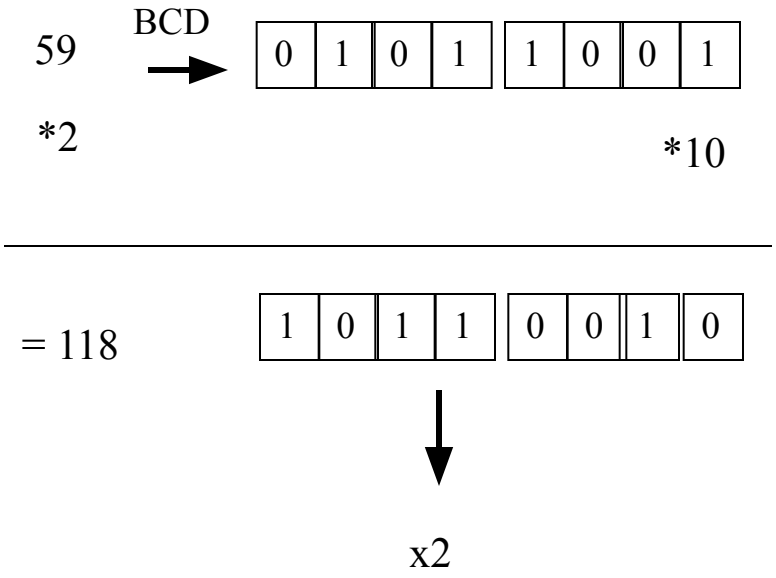
Операция **Образование дополнения до единицы для дв. слова**



Операнды : IN: VD, ED, AD, MD, SMD, AC, HC, константа , *VD, *AC, SD
OUT: VD, ED, AD, MD, SMD, AC, *VD,*AC, SD

Необходимость преобразований

Необходимо умножить какое либо число, например 59 на 2
результат соответственно должен быть равен 118.
Но если число 59 представлено в BCD то получим неверный результат.

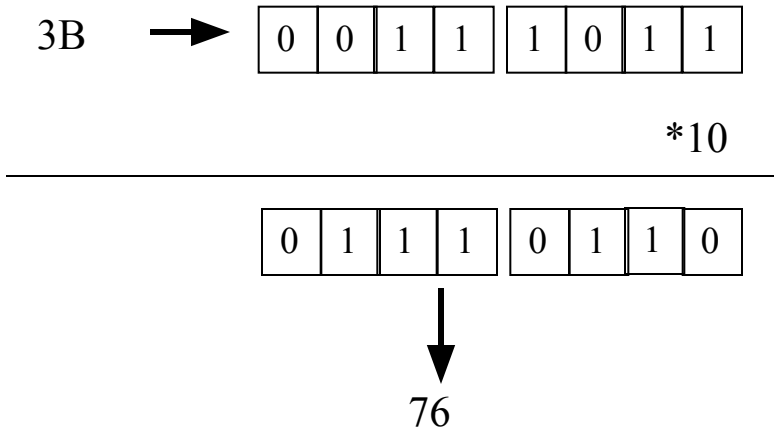


Т.о, вместо 118 в результате получилось x2.

Поэтому, прежде чем проводить операцию умножения, необходимо выполнить ряд преобразований:

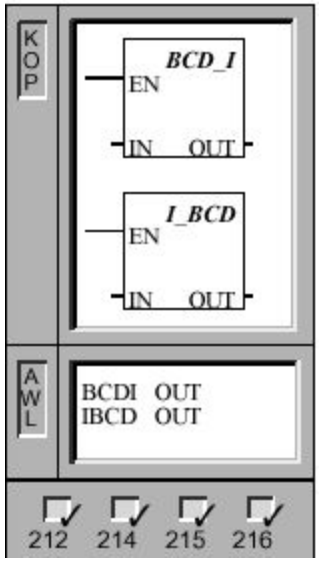
1). BCD число перевести в целое(т.е в 16-ричное)
59 (BCD) → 3B(Hex)

2). Произвести операцию умножения



3). Полученное целое число перевести в BCD:
76(Hex) → 7*16+6*1 = 112 + 6 = 118

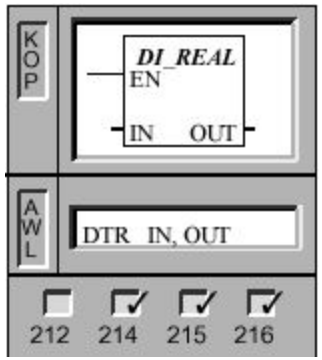
Операции преобразования



Операция **Преобразование BCD в целое число** преобразует двоично - десятичное значение (IN) в целочисленное значение и загружает результат в OUT.

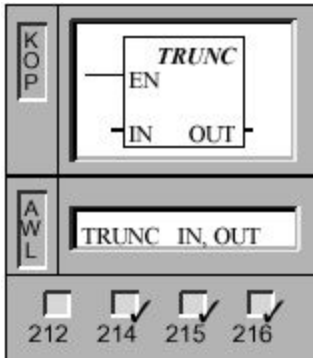
Операция **Преобразование целого числа в BCD** преобразует целочисленное значение (IN) в двоично - десятичное значение и загружает результат в OUT.

Операнды : IN: VW, T, Z, EW, AW, MW, SMW, AC, AEW, константа , *VD, *AC, SW.
OUT: VW, T, Z, EW, AW, MW, SMW, AC, *VD, *AC, SW.



Операция **Преобразование целого числа (32 бита) в действительное число** преобразует целое число (32 бита) со знаком (IN) в действительное число (32 бита) (OUT).

Операнды : IN: VD, ED, AD, MD, SMD, AC, HC, константа , *VD, *AC, SD
OUT: VD, ED, AD, MD, SMD, AC, *VD, *AC, SD



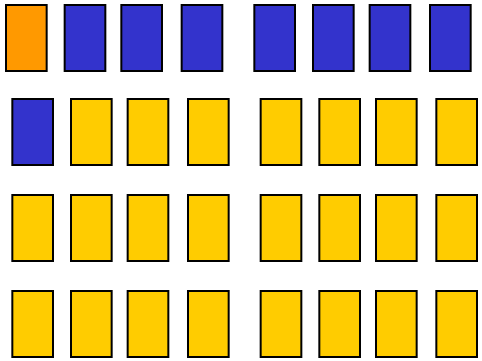
Операция **Преобразование действительного числа в целое число (32 бита)** преобразует действительное число (IN) в целое число (32 бита) (OUT). Преобразуется только целочисленная часть действительного числа (отбрасыванием знаков после десятичной точки).

Операнды : IN: VD, ED, AD, MD, SMD, AC, HC, константа , VD, *AC
OUT: VD, ED, AD, MD, SMD, AC, *VD, *AC

Формат действительного числа

Размер действительного числа составляет двойное слово. Само число представлено в виде:

$(1 + M) * 2^{(E - 127)}$.



- знак числа: «0»- положительное
«1»- отрицательное

- степень основания 2

0111 1101	-2
0111 1110	-1
0111 1111	0
1000 0000	1
1000 0001	2
1000 0010	3

и т.д

- мантисса M, определяемая суммой 1 и коэффициентов значащих разрядов: 1/2, 1/4, 1/8, 1/16 и т.д

VB0	0	1	0	0	0	0	0	0
VB1	0	1	1	0	0	0	0	0
VB2	0	0	0	0	0	0	0	0
VB3	0	0	0	0	0	0	0	0

- Положительное число;
- E = 1;
- M= 1+ 1/2+ 1/4= 7/4

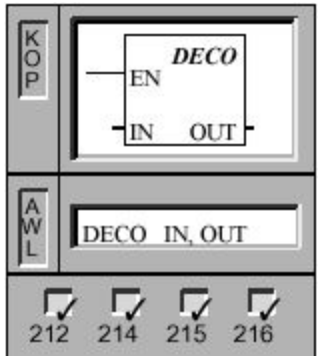
VD0 $7/4 * 2^{1} = 3,5$

VB0	1	0	1	1	1	1	1	0
VB1	1	1	1	0	0	0	0	0
VB2	0	0	0	0	0	0	0	0
VB3	0	0	0	0	0	0	0	0

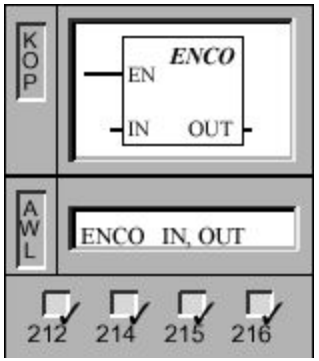
- Отрицательное число;
- E = -2;
- M= 1+ 1/2+ 1/4= 7/4

VD0 $7/4 * 2^{-2} = 0,4375$

Операции преобразования



Операция **Преобразование бита в шестнадцатиричное число** устанавливает в выходном слове (OUT) бит , номер которого (бит #) соответствует тому , который представлен младшим полубайтом (4 бита входного байта (IN). Остальные биты выходного слова устанавливаются “0”.

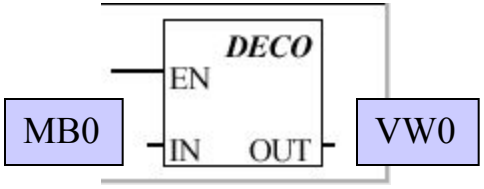


Операция **Преобразование шестнадцатиричного числа в бит** записывает номер младшего значащего бита (бит #) входного слова (IN) в младший полубайт (4 бита) выходного байта (OUT).

VW0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0

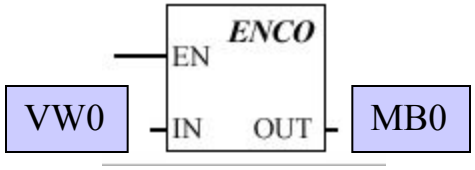
MB0	0	0	0	0	0	0	1	1

3



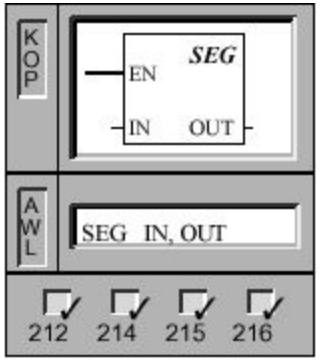
VW0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

10



MB0	0	0	0	0	1	0	1	0

Операции преобразования

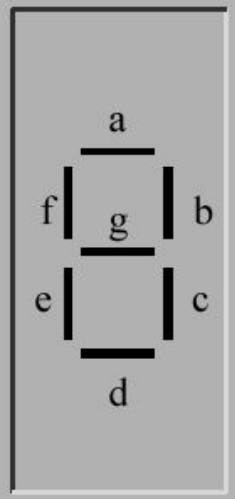


Операция **Образование битовой комбинации для семисегментного индикатора** образует битовую комбинацию (OUT), которая подсвечивает сегменты семисегментного индикатора . Подсвечиваемые сегменты представляют знак младшей цифры входного байта (IN).

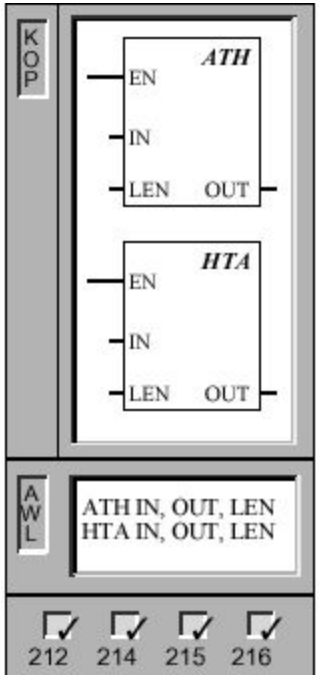
Операнды : IN: VB, EB, AB, MB, SMB, AC, константа , *VD, *AC, SB

OUT: VB, EB, AB, MB, SMB, AC,*VD, *AC, SB

(IN) LSD	Отображ. сегментов	(OUT) - g f e d c b a			Отображ. сегментов	(OUT) - g f e d c b a
0		0 0 1 1 1 1 1 1		8		0 1 1 1 1 1 1 1
1		0 0 0 0 0 1 1 0		9		0 1 1 0 0 1
2		0 1 0 1 1 0 1 1		A		0 1 1 1 0 1 1 1
3		0 1 0 0 1 1 1 1		B		0 1 1 1 1 1 0 0
4		0 1 1 0 0 1 1 0		C		0 0 1 1 1 0 0 1
5		0 1 1 0 1 1 0 1		D		0 1 0 1 1 1 1 0
6		0 1 1 1 1 1 0 1		E		0 1 1 1 1 0 0 1
7		0 0 0 0 0 1 1 1		F		0 1 1 1 0 0 0 1



Операции преобразования



Операция **Преобразование строки символов ASCII–кода в шестнадцатиричное число** Преобразует строку длиной LEN символов ,начиная с символа IN, в шестнадцатиричные цифры , которые начинаются с адреса OUT. Строка символов может быть длиной максимум 255 символов .

Операция **Преобразование шестнадцатиричного числа в строку символов ASCII–кода** преобразует шестнадцатиричные цифры , начиная с входного байта (IN), в строку символов ASCII–кода , которая начинается с адреса OUT. Количество шестнадцатиричных цифр , подлежащих преобразованию , задается длиной (LEN). Можно преобразовать максимум 255 шестнадцатиричных цифр .

Операнды : IN, OUT: VB, EB, AB, MB, SMB, *VD, *AC, SB
LEN: VB, EB, AB, MB, SMB, AC, константа , *VD, *AC, SB
Допустимыми ASCII–символами являются шестнадцатиричные значения от 30 до 39 и от 41 до 46.

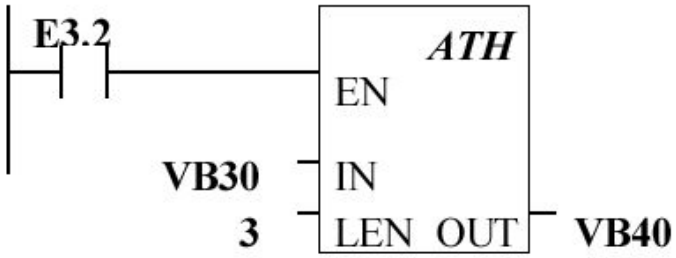
ASCII - Американский стандартный код для обмена информацией является стандартом для представления алфавитно -цифровых данных.

Расширенные коды ASCII:
Знаки препинания:
21 - ! 2A - *
22 - « 2B - +
23 - # 2C - ,
24 - $ 2D - -
25 - % 2E - .
26 - & 2F - /
27 -’
28 - (
29 -)

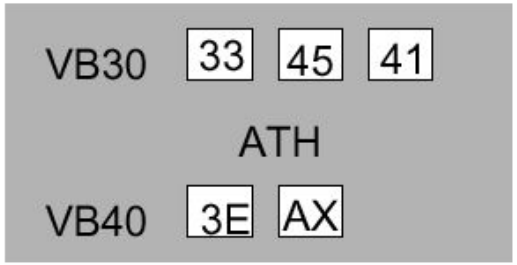
Расширенные коды ASCII:
Цифры и знаки:
30 - 0
31 - 1 3A - :
32 - 2 3B - :
33 - 3 3C - <
34 - 4 3D - =
35 - 5 3E - .>
36 - 6 3F - ?
37 -7
38 - 8
39 - 9

Расширенные коды ASCII:
Буквы:
40 - @ 50 - P
41 - A 4A - J 51 - Q 5A -Z
42 - B 4B - K 52 - R 5B -[
43 - C 4C - L 53 - S 5C -
44 - D 4D - M 54 - T 5D -]
45 - E 4E - N 55 - U 5E - ^
46 - F 4F - O 56 - V 5F - -
47 -G 57 - W Прописные
48 - H 58 - X соответственно
49 - I 59 - Y начиная с 60.

Пример преобразования

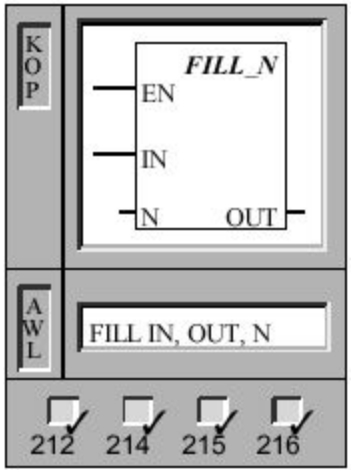
KOP	AWL
	LD E3.2 ATH VB30, VB40, 3

Применение



Указание: X указывает, что полубайт не был изменен.

Заполнение памяти

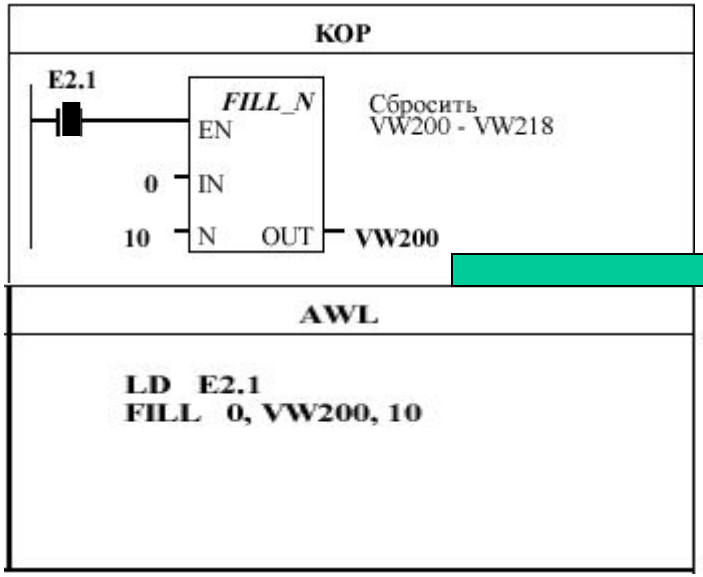


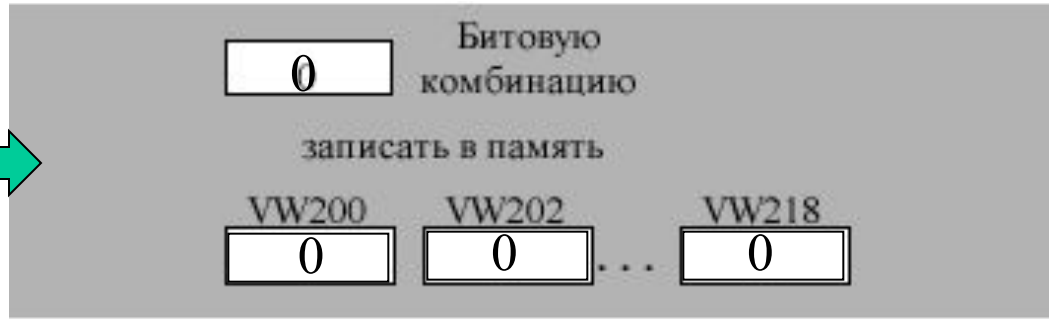
Операция **Заполнение памяти битовой комбинацией** заполняет область памяти , начинающейся с выходного слова OUT, битовой комбинацией входного слова IN для заданного количества слов N.
N может лежать в диапазоне от 1 до 255.

Операнды :IN: VW, T, Z, EW, AW, MW, SMW, AEW, константа , *VD, *AC, SW

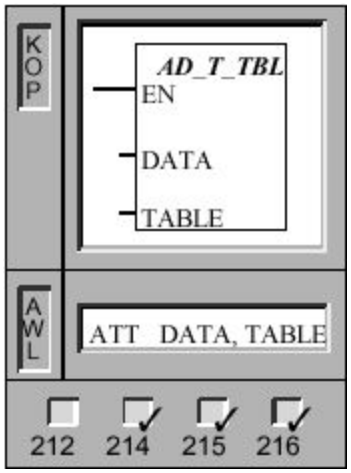
OUT: VW, T, Z, EW, AW, MW,SMW, AAW,*VD, *AC, SW

N: VB, EB, AB, MB, SMB, AC,константа , *VD, *AC, SB





Запись значения в таблицу

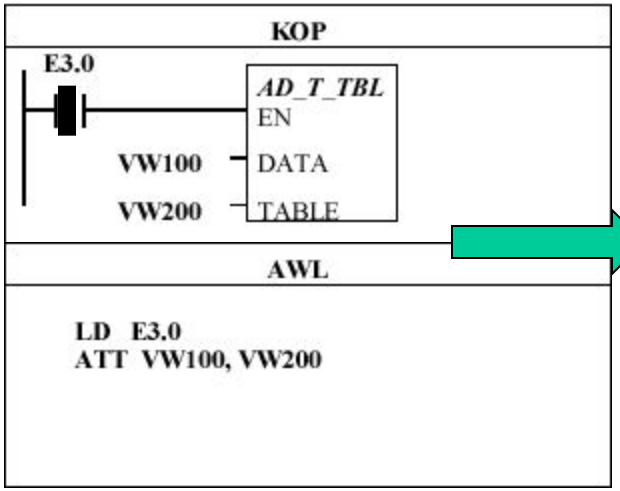


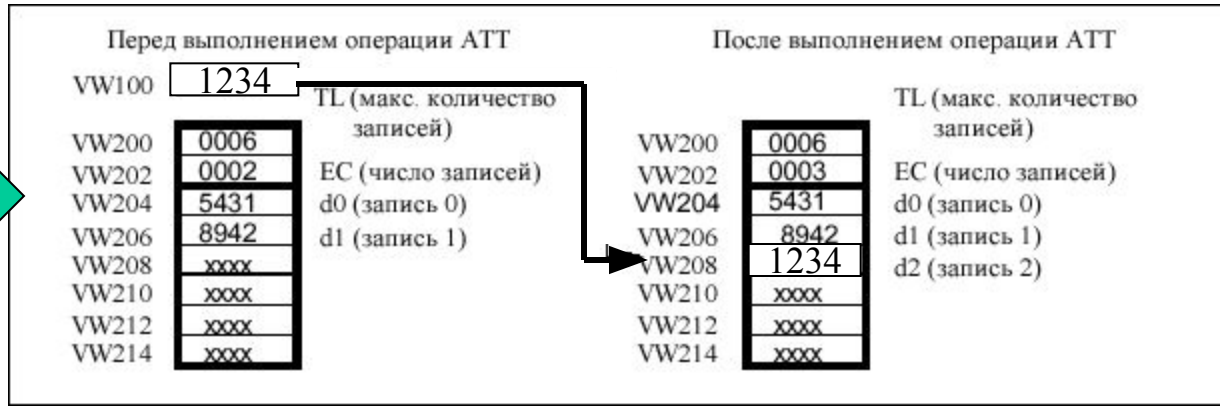
Операция **Запись значения в таблицу** вносит значения слов (DATA) в таблицу (TABLE).

Операнды : DATA: VW, T, Z, EW, AW, MW, SMW, AC,AEW, константа , *VD, *AC, SW

TABLE: VW, T, Z, EW, AW, MW, SMW, *VD, *AC,SW

Первое значение в таблице задает максимальную длину таблицы (TL). Второе значение задает количество записей в таблице (EC). Новые данные добавляются в таблице после последней записи . Каждый раз , когда записываются новые данные , количество записей увеличивается на “1”. Таблица может содержать максимум 100 записей , исключая параметры , задающие максимальную длину таблицы и фактическое количество записей .

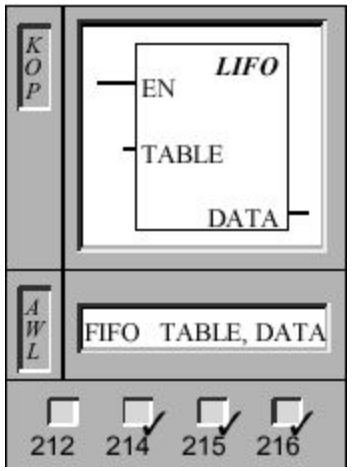




Эта операция влияет на следующие специальные меркеры :

SM1.4 устанавливается в “1”, если Вы пытаетесь записать в таблицу слишком много значений .

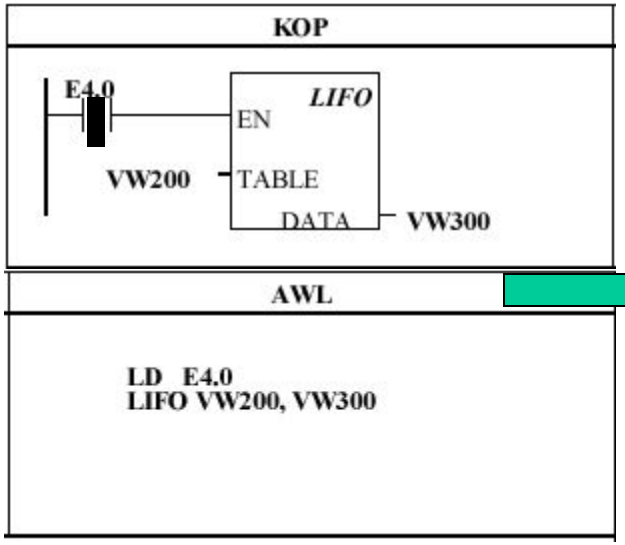
L I F O

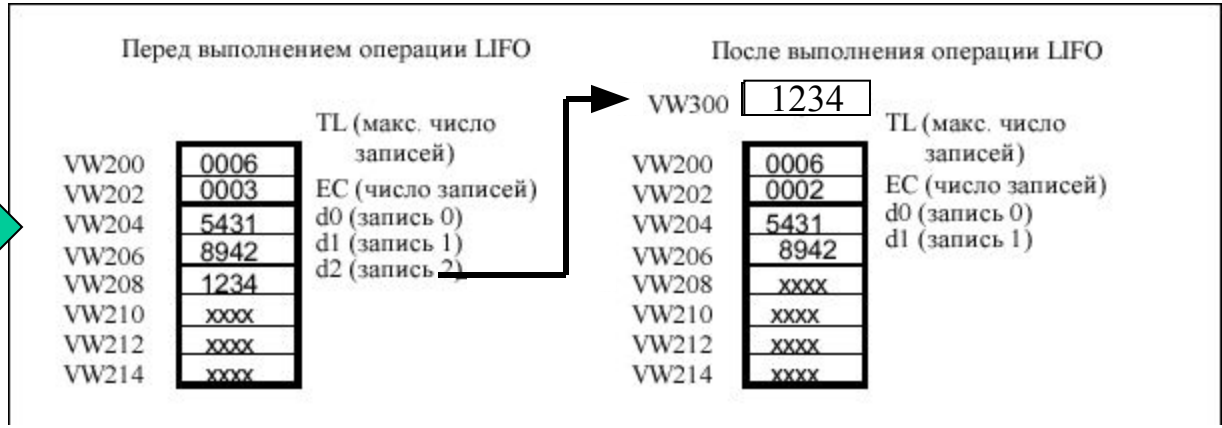


Операция **Стирание последней записи в таблице (LIFO)** стирает последнюю запись в таблице (TABLE) и выводит значение по адресу DATA. Каждый раз , когда выполняется данная операция , количество записей (EC) уменьшается на “1”.

Операнды :TABLE: VW, T, Z, EW, AW, MW,SMW, *VD, *AC, SW

DATA: VW, T, Z, EW, AW, MW,SMW, AC, AAW, *VD,*AC, SW

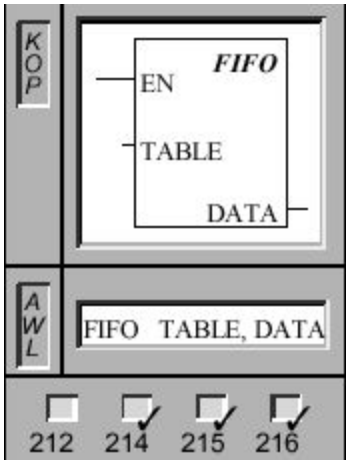




Эта операция влияет на следующие специальные меркеры :

SM1.5 (устанавливается в “1”, если Вы пытаетесь стереть запись в пустой таблице).

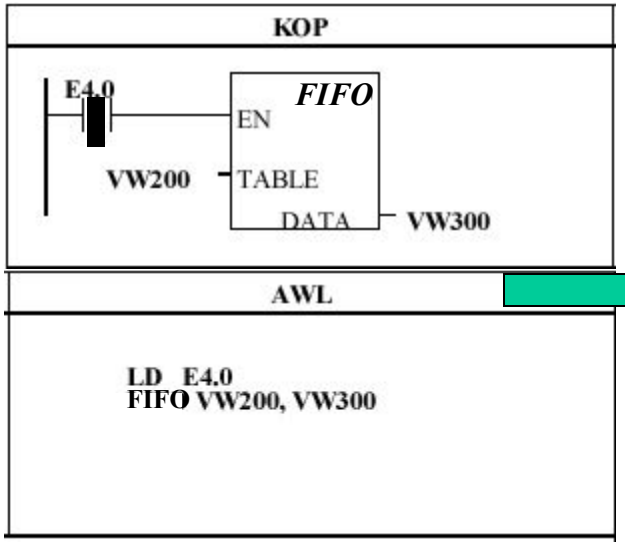
F I F O

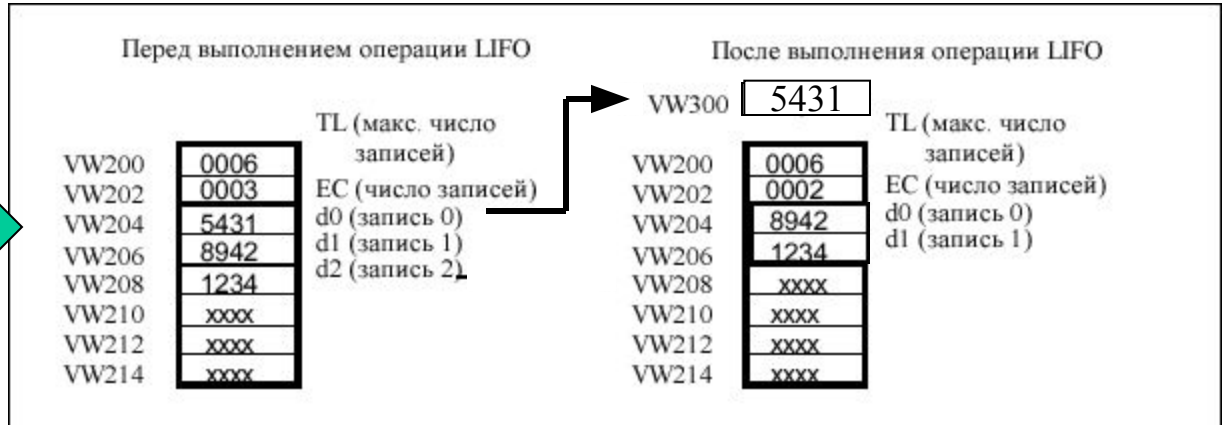


Операция **Стирание первой записи в таблице (FIFO)** стирает первую запись в таблице (TABLE) и выводит значение по адресу DATA. Каждый раз , когда выполняется данная операция , количество записей (EC) уменьшается на “1”.

Операнды :TABLE: VW, T, Z, EW, AW, MW,SMW, *VD, *AC, SW

DATA: VW, T, Z, EW, AW, MW,SMW, AC, AAW, *VD,*AC, SW



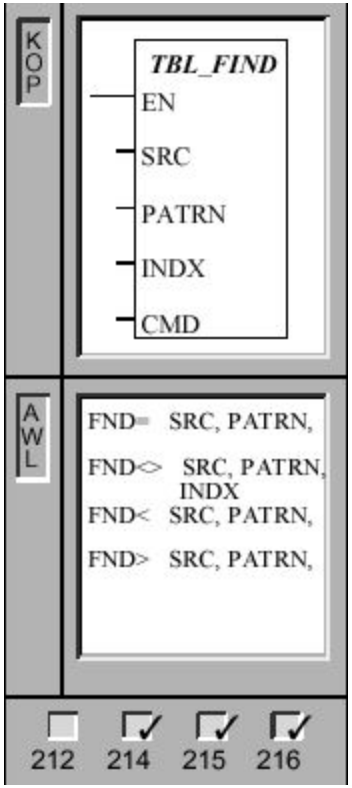


Эта операция влияет на следующие специальные меркеры :

SM1.5 (устанавливается в “1”, если Вы пытаетесь стереть запись в пустой таблице).

Поиск значения в таблице

Операция **Поиск значения в таблице** просматривает таблицу (SRC),начиная с записи таблицы , заданной параметром INDX, в поисках значения данных (PATRN), соответствующего заданным критериям =, <>,< или >.
В KOP параметр CMD задает критерий числовым значением от 1 до 4, что соответствует критерию =, <>, < или >.
Операнды :SRC: VW, T, Z, EW, AW, MW, SMW, *VD, *AC, SW
PATRN: VW, T, Z, EW, AW, MW, SMW, AC, AEW, константа , *VD, *AC, SW
INDX: VW, T, Z, EW, AW, MW, SMW, AC, *VD, *AC, SW
CMD: 1 (=) 2 (<>) 3 (<) 4 (>)
Если соответствующая запись в таблице найдется , то INDX указывает эту запись . Если в таблице нет подходящей записи , то значение INDX соответствует количеству записей в таблице . Для того , чтобы искать следующую запись , нужно сначала увеличить INDX на “1”. Лишь тогда операция может быть вызвана снова. Записи в таблице (где должен производиться поиск) пронумерованы от 0 до максимального значения . Таблица может содержать максимум 100 записей . Сюда не включены параметры , задающие максимальную длину таблицы и фактическое количество записей в таблице .



Указание
Если Вы используете операции поиска в таблицах , составленных с помощью операций ATT, LIFO и FIFO, то количество записей и записи данных имеют прямое соответствие . В отличие от операций ATT, LIFO и FIFO, где максимальное количество записей задается в слове , операции поиска не используют данное слово . Поэтому операнд SRC операции поиска располагается на один адрес слова (байта) выше , чем операнд таблицы , соответствующей операции ATT, LIFO или FIFO, как показано на рисунке:

Формат таблицы для ATT, LIFO и FIFO

VW200	0006	TL (макс. число записей)
VW202	0006	EC (число записей)
VW204	xxxx	d0 (данные 0)
VW206	xxxx	d1 (данные 1)
VW208	xxxx	d2 (данные 2)
VW210	xxxx	d3 (данные 3)
VW212	xxxx	d4 (данные 4)
VW214	xxxx	d5 (данные 5)

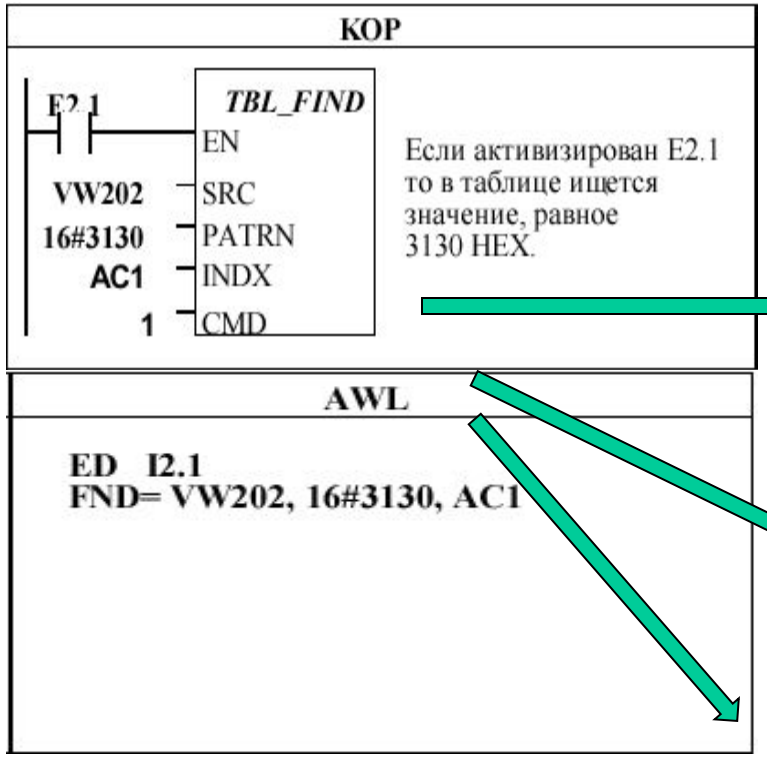
Формат таблицы для TBL_FIND

VW202	0006	TL (макс. число
VW204	xxxx	d0 (данные 0)
VW206	xxxx	d1 (данные 1)
VW208	xxxx	d2 (данные 2)
VW210	xxxx	d3 (данные 3)
VW212	xxxx	d4 (данные 4)
VW214	xxxx	d5 (данные 5)

Пример операции поиска

Это таблица , которую Вы просматриваете . Если таблица была создана с помощью операции ATT, LIFO или FIFO, то VW200 содержит максимально допустимое число записей и не требуется для операций поиска .

VW202	0006	EC (число записей)
VW204	3133	d0 (запись 0)
VW206	4142	d1 (запись 1)
VW208	3130	d2 (запись 2)
VW210	3030	d3 (запись 3)
VW212	3130	d4 (запись 4)
VW214	4541	d5 (запись 5)



AC 1	0	AC1 нужно сбросить в "0", чтобы вести поиск с самой верхней записи таблицы .
Поиск		
AC 1	2	AC1 содержит номер первой записи , соответствующей критерию поиска .
AC 1	3	Увеличить INDX на "1" перед просмотром остальных записей таблицы .
Поиск		
AC 1	4	AC1 содержит номер второй записи , соответствующей критерию поиска .
AC 1	5	Увеличить INDX на "1" перед просмотром остальных записей таблицы .
Поиск		
AC 1	6	AC1 содержит значение , соответствующее числу записей в таблице . Вся таблица просмотрена , дальнейшие подходящие записи не найдены .
AC 1	0	Для получения возможности нового поиска в таблице нужно сбросить значение INDX в "0".