Логические операции

Логические операции выполняются поразрядно, то есть отдельно для каждого бита операндов. В результате выполнения изменяются флаги. В программах эти операции часто используются для сброса, установки или инверсии отдельных битов двоичных чисел.

Логическое И
Если оба бита равны 1, то результат равен 1, иначе результат равен 0.

AND	**0**	**1**
0	0	0
1	0	1

Для выполнения операции логического И предназначена команда AND. У этой команды 2 операнда, результат помещается на место первого операнда. Часто эта команда используется для обнуления определённых битов числа. При этом второй операнд называют *маской*. Обнуляются те биты операнда, которые в маске равны 0, значения остальных битов сохраняются.

Примеры:

```
and ax,bx          ;AX = AX & BX
and cl,11111110b   ;Обнуление младшего бита CL
and dl,00001111b   ;Обнуление старшей тетрады DL
```

Ещё одно использование этой команды — быстрое вычисление остатка от деления на степень 2. Например, так можно вычислить остаток от деления на 8:

```
and ax,111b        ;AX = остаток от деления AX на 8
```

Логическое ИЛИ
Если хотя бы один из битов равен 1, то результат равен 1, иначе результат равен 0.

OR	**0**	**1**
0	0	1
1	1	1

Логическое ИЛИ вычисляется с помощью команды OR. У этой команды тоже 2 операнда, и результат помещается на место первого. Часто эта команда используется для установки в 1 определённых битов числа. Если бит маски равен 1, то бит результата будет равен 1, остальные биты сохранят свои значения.

Примеры:

```
or al,dl           ;AL = AL | DL
or bl,10000000b    ;Установить знаковый бит BL
or cl,00100101b    ;Включить биты 0,2,5 CL
```

Логическое НЕ (инверсия)
Каждый бит операнда меняет своё значение на противоположное (0→1, 1→0). Операция выполняется с помощью команды NOT. У этой команды только один операнд. Результат помещается на место операнда. Эта команда не изменяет значения флагов.

Пример:

```
not byte[bx]       ;Инверсия байта по адресу в BX
```

Логическое исключающее ИЛИ (сумма по модулю два)

Если биты имеют одинаковое значение, то результат равен 0, иначе результат равен 1.

XOR	0	1
0	0	1
1	1	0

Исключающим ИЛИ эта операция называется потому, что результат равен 1, если один бит равен 1 или другой равен 1, а случай, когда оба равны 1, исключается. Ещё эта операция напоминает сложение, но в пределах одного бита, без переноса. 1+1=10, но перенос в другой разряд игнорируется и получается 0, отсюда название «сумма по модулю 2». Для выполнения этой операции предназначена команда XOR. У команды два операнда, результат помещается на место первого. Команду можно использовать для инверсии определённых битов операнда. Инвертируются те биты, которые в маске равны 1, остальные сохраняют своё значение.

Примеры:

```
xor si,di           ;SI = SI ^ DI
xor al,11110000b    ;Инверсия старшей тетрады AL
xor bp,8000h        ;Инверсия знакового бита BP
```

Обозначение операции в комментарии к первой строке используется во многих языках высокого уровня (например C, C++, Java и т.д.). Часто XOR используют для обнуления регистров. Если операнды равны, то результат операции всегда равен 0. Такой способ обнуления работает быстрее и, в отличие от команды MOV, не содержит непосредственного операнда, поэтому команда получается короче (и не содержит нулевых байтов):

```
mov bx,0        ;Эта команда занимает 3 байта
xor bx,bx       ;А эта — всего 2 байта
```

Пример программы

Допустим, у нас есть массив байтов. Размер массива хранится в байте без знака. Требуется в каждом байте сбросить 1-й и 5-й биты, установить 0-й и 3-й биты, инвертировать 7-й бит. А затем ещё инвертировать целиком последний байт массива.

```
use16                   ;Генерировать 16-битный код
org 100h                ;Программа начинается с адреса 100h

    mov bx,array        ;BX = адрес массива
    movzx cx,[length]   ;CX = длина массива

    mov di,cx
    dec di
    add di,bx           ;DI = адрес последнего элемента
m1:
    mov al,[bx]         ;AL = очередной элемент массива
    and al,11011101b    ;Сбрасываем 1-й и 5-й биты
    or  al,00001001b    ;Устанавливаем 0-й и 3-й биты
    xor al,10000000b    ;Инвертируем 7-й бит
    mov [bx],al         ;Сохраняем обработанный элемент
    inc bx              ;В BX - адрес следующего элемента
    loop m1             ;Команда цикла

    not byte[di]        ;Инвертируем последний байт массива
```

```
    mov ax,4C00h        ;\
    int 21h             ;/ Завершение программы
;-------------------------------------------------------
length  db 10
array   db 1,5,3,88,128,97,253,192,138,0
```

Задание.
Объявите переменную *x* как двойное слово с каким-то значением.
Инвертируйте 7-й, 15-й и 31-й бит.
Обнулите младший байт переменной.
Присвойте единичное значение битам 11-14 и 28-30.
Результат сохраните в переменной *y* (она тоже должна быть объявлена как двойное слово).
Инвертируйте значение *x*.