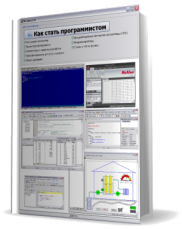

Как стать программистом

Бесплатная книга о программировании
для начинающих и бывалых.

Получить >>>

Подписаться:

Микроконтроллеры для ЧАЙНИКОВ
Изучать БЕСПЛАТНО

14.09.2020 г.
Добавлена статья Уменьшение энергопотребления.

05.09.2020 г.
Добавлены видео и статья Самое простое устройство на микроконтроллере.

21.08.2020 г.
Добавлены видео и статья Инструкция CLI.

19.06.2020 г.
Добавлена статья Выводы ATtiny13A.

19.05.2020 г.
Добавлена статья Регистр PRR.

Команда XOR

Что такое JavaScript

Если вы интересуетесь программированием вообще, и сайтостроением в частности, то вы наверняка слышали слово JavaScript. И, если вы до сих пор не узнали толком, что же это такое, то пришло время сделать это. Подробнее...

Команда XOR в Ассемблере выполняет операцию исключающего ИЛИ между всеми битами двух операндов. Результат операции XOR записывается в первый операнд. Синтаксис:

```
XOR ПРИЁМНИК, ИСТОЧНИК
```

Инструкция XOR всегда сбрасывает флаги CF и OF, а также (в зависимости от результата) изменяет флаги SF, ZF и PF. Значение флага AF может быть любым - оно не зависит от результата операции.

ПРИЁМНИК может быть одним из следующих:

- Область памяти (MEM)
- Регистр общего назначения (REG)

ИСТОЧНИК может быть одним из следующих:

- Область памяти (MEM)
- Регистр общего назначения (REG)
- Непосредственное значение - константа (IMM)

С учётом ограничений, которые были описаны выше, комбинации ПРИЁМНИК-ИСТОЧНИК могут быть следующими:

```
REG,            MEM
MEM,            REG
REG,            REG
MEM,            IMM
REG,            IMM
```

Операция исключающего ИЛИ

При выполнении операции исключающего ИЛИ значение результата будет равно 1, если сравниваемые биты отличаются (не равны). Если же сравниваемые биты имеют одинаковое значение, то результат будет равен 0.

Потому эта операция и называется исключающей. Она исключает из сравнения одинаковые биты, а с неодинаковыми выполняет операцию логического ИЛИ.

Но, так как любая пара неодинаковых битов это 0 и 1, то операция логического ИЛИ в результате даст 1.

Таблица истинности исключающего ИЛИ

Таблица истинности XOR приведена ниже:

```
0     XOR     0     =     0
0     XOR     1     =     1
1     XOR     0     =     1
1     XOR     1     =     0
```

Особенности операции XOR

Операция XOR обладает свойством реверсивности. Если её выполнить дважды с одним и тем же операндом, то значение результата инвертируется. То есть если два раза выполнить эту операцию между битами **X** и **Y**, то в конечном результате мы получим исходное значение бита **X**.

```
0     XOR     0     =     0     XOR     0     =     0
0     XOR     1     =     1     XOR     1     =     0
1     XOR     0     =     1     XOR     0     =     1
1     XOR     1     =     0     XOR     1     =     1
```

Это свойство можно использовать, например, для простейшего шифрования данных (об этом как-нибудь в другой раз).

Проверка флага чётности после операции XOR

Команда XOR работает с 8-, 16- и 32-разрядными операциями.

Иногда есть необходимость после выполнения операции проверить флаг чётности PF, для того, чтобы узнать, какое количество единичных битов (чётное или нечётное) содержится в **младшем байте**

результата (это бывает необходимо не только в случае выполнения операции XOR, но и при выполнении других арифметических и логических операций).

Если флаг чётности установлен, то в результате получилось чётное количество единичных битов. Иначе флаг будет сброшен.

Можно также просто проверить на чётность любое число, не меняя значения результата. Для этого надо выполнить команду XOR с нулевым значением. То есть в ПРИЁМНИКЕ должно быть проверяемое число, а в ИСТОЧНИКЕ должен быть ноль. А затем надо проверить флаг чётности. Пример:

```
MOV  AL,  10110101b            ;Поместить в AL число с нечётным
                               ;количеством единичных битов (5)
XOR  AL,  0                    ;При этом флаг чётности PF не
                               ;устанавливается (PO)
MOV  AL,  10110111b            ;Поместить в AL число с чётным
                               ;количеством единичных битов (6)
XOR  AL,  0                    ;При этом флаг чётности PF
                               ;будет установлен (PE)
```

В отладчиках обычно для обозначения чётного количества единиц в полученном результате используется сокращение PE (Parity Even), а для нечётного - PO (Parity Odd).

Чётность в 16-разрядных словах

Как уже было сказано, флаг чётности устанавливается в зависимости от количества единиц, содержащихся в младшем байте результата. Чтобы проверить чётность 16-разрядного операнда, надо выполнить команду XOR между старшим и младшим байтом этого числа:

```
MOV  AX,  64C1h          ;0110 0100 1100 0001 - 6 единичных битов
XOR  AH,  AL             ;Флаг чётности будет установлен
```

Таким нехитрым способом 16-разрядный операнд разбивается на два байта (2 группы по 8 битов), и при выполнении команды XOR единичные биты, находящиеся в соответствующих разрядах двух 8-разрядных операндов, не будут учитываться. Потому что соответствующий бит результата равен нулю.

Команда XOR удаляет из результата любые пересекающиеся единичные биты двух 8-разрядных операндов и добавляет в результат непересекающиеся единичные биты. То есть чётность полученного нами 8-разрядного числа будет такой же, как и чётность исходного 16-разрядного числа.

```
0110 0100 1100 0001 - исходное 16-разрядное число

0     XOR     1     =     1
1     XOR     1     =     0
1     XOR     0     =     1
0     XOR     0     =     0
0     XOR     0     =     0
1     XOR     0     =     1
0     XOR     0     =     0
0     XOR     1     =     1
```

В результате 4 единицы, то есть флаг PF будет установлен

Чётность в 32-разрядных двойных словах

Ну а если надо определить чётность в 32-разрядном числе?

Тогда число разбивается на четыре байта, и поочерёдно с этими байтами выполняется операция исключающего ИЛИ.

Например, мы разбили 32-разрядное число **B** на четыре байта **B0**, **B1**, **B2**, **B3**, где **B0** - это младший байт.

Тогда для определения чётности числа B нам надо будет использовать следующую формулу:

```
B0 XOR B1 XOR B2 XOR B3
```

Но в ассемблере такая запись недопустима. Поэтому придётся немного подумать.

Ну и напоследок о происхождении мнемоники **XOR**. В английском языке есть слово e**X**ception - исключение. Сокращением от этого слова является буква **X** (так повелось). Вы наверняка встречали такое в рекламе или в названии продуктов, производители которых претендуют (ну или думают, что претендуют) на исключительность. Например, Лада XRAY, Sony XPeria и т.п. Так что XOR - это аббревиатура, собранная из двух слов - e**X**ception **OR** - исключающее ИЛИ.