Основы C++

Неважно, на каком языке вы программируете. Если вы не знаете C++, вы не можете считать себя программистом.

Подробнее >>>

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.

Инструкция AND

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...

ПРИМЕЧАНИЕ
В видео я немного попутал нумерацию разрядов. Но, надеюсь, вас это не расстроило)))

Инструкция AND выполняет логическое И между всеми битами двух операндов. Результат записывается в первый операнд. Синтаксис:

```
AND ЧИСЛО1, ЧИСЛО2
```

ЧИСЛО1 может быть одним из следующих:

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

ЧИСЛО2 может быть одним из следующих:

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

С учётом ограничений, которые были описаны выше, комбинации ЧИСЛО1-ЧИСЛО2 могут быть следующими:

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

Как было сказано, команда AND выполняет операцию логического И между всеми битами двух чисел. Таблица истинности для операции логического И выглядит следующим образом:

```
0       И       0       =       0
0       И       1       =       0
1       И       0       =       0
1       И       1       =       1
```

Почему это называется таблицей истинности? Потому что она отображает результаты выполнения функции с разными параметрами. И эти результаты могут принимать только одно из двух значений: ИСТИНА (логическая 1) или ЛОЖЬ (логический 0).

Логическое И - это операция логического умножения. Чтобы легче было запомнить таблицу истинности для логического И, вспомните математику:

```
0 * 0 = 0
0 * 1 = 0
1 * 0 = 0
1 * 1 = 1
```

Что такое битовая маска

Для чего используются логические операции, такие как логическое умножение? И зачем в ассемблере команды, такие как AND?

Дело в том, что довольно часто приходится сравнивать какое-то число с так называемой битовой маской. И в зависимости от результата выполнять какие-то действия.

Иными словами, довольно часто бывает так, что надо узнать состояние отдельного бита (или нескольких битов) в числе.

Например, нерационально тратить целый байт, чтобы хранить там какое-то логическое значение (например, сигнал на каком-то входе процессора). Поэтому в таких случаях обычно упаковывают в байт 8 логических сигналов, где каждому отдельному биту соответствует какое-то логическое значение.

Давайте представим, что у нас есть какое-то устройство с 8 входами. И сигналы с этих входов каким-то образом упаковываются в один байт, где каждому биту соответствует отдельный вход (с 0 до 7).

А теперь вопрос - как нам узнать, есть сигнал на входе 3 или нет? То есть установлен третий бит в 1, или сброшен в 0?

А вот для этого-то и существуют битовые маски.

Если нам надо, например, узнать состояние третьего входа (отсчёт с нуля, справа налево), то битовая маска для проверки состояния 3-го бита будет такой:

```
00001000
```

Теперь с помощью логических операций мы можем сравнивать число с этой битовой маской и получать результат.

Делается это просто - если после выполнения операции логического И результат будет нулевым, то интересующий нас бит сброшен (то есть равен 0). Если же результат будет НЕ нулевым (равным маске), то бит установлен (равен 1).

Давайте попробуем сравнить число 150 (10010110 в двоичной системе) с нашей битовой маской. Как видите, в числе 150 третий бит сброшен. Итак (вверху расположен 7-й бит):

```
Число    Маска
1     И  0     = 0
0     И  0     = 0
0     И  0     = 0
1     И  0     = 0
0     И  1     = 0
1     И  0     = 0
1     И  0     = 0
0     И  0     = 0
```

Результат равен 0, поэтому делаем вывод, что бит 3 сброшен, то есть также равен 0.

А теперь с числом 159 (10011111)

```
Число    Маска
1     И  0     = 0
0     И  0     = 0
0     И  0     = 0
1     И  0     = 0
1     И  1     = 1
1     И  0     = 0
1     И  0     = 0
1     И  0     = 0
```

Результат НЕ равен 0, поэтому делаем вывод, что бит 3 установлен, то есть равен 1.

После выполнения команды AND можно, например, выполнить уже известную нам команду CMP, а затем с помощью какой-либо из инструкций условного перехода направить выполнение программы по нужному нам руслу.

К примеру, если вы создаёте программу для микроконтроллера, то вы можете прочитать состояние портов ввода, а для дальнейшей обработки упаковать их в байт (дальнейшая обработка может понадобиться в тех случаях, например, когда работа устройства зависит от комбинации состояний на входах).

Затем вы сравниваете байт с битовой маской, и в зависимости от результата предпринимаете необходимые действия.

Инструкция AND может применяться и для других целей. Например, вот этот код:

```
MOV AL, 'a'        ; AL = 01100001b
AND AL, 11011111b  ; AL = 01000001b  ('A')
```

преобразует маленькую букву a в большую.

Пример использования инструкции AND:

```
.model  tiny
.code
ORG     100h

start:

MOV AX, 0B800h    ; установить AX = B800h (память VGA).
MOV DS, AX        ; копировать значение из AX в DS.
MOV CL, 'a'       ; CL = 61h (ASCII-код символа 'a').
MOV CH, 01001110b ; CH = атрибуты цвета (желтый текст на красном фоне).
MOV BX, 72eh      ; BX = позиция на экране = 2*(x + y*80) = (39, 11).
MOV [BX], CX      ; [0B800h:015Eh] = CX (записать символ в видеопамять).
AND CL, 01000001b ; Теперь CL = 41h (ASCII-код символа 'A').
MOV BX, 730h      ; BX = позиция на экране = 2*(x + y*80) = (40, 11).
MOV [BX], CX      ; [0B800h:015Eh] = CX (записать символ в видеопамять).

END     start
```

Как работает инструкция MOV в данном примере, подробно рассказано здесь.

Обычно я привожу расшифровку аббревиатуры инструкции. Однако в данном случае это не требуется, так как **AND** переводится как **И**.