



Что такое JavaScript

Вы наверняка слышали слово JavaScript. И, если вы до сих пор не узнали толком, что же это такое, то пришло время сделать это. [Подробнее >>>](#)

Подписаться:



[Главная](#) [Ассемблер](#) [Микроконтроллеры](#) [Инструкции Intel](#) [Дневник](#)



Микрокон
[Изучать Б](#)

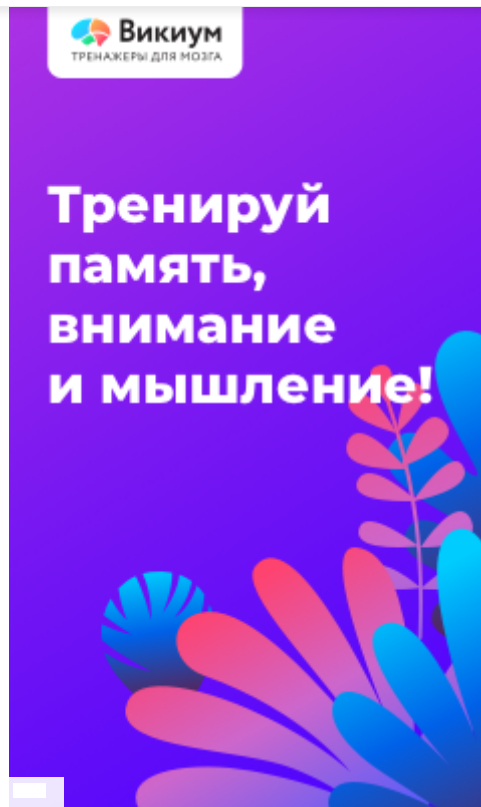
http://av-assembler.ru запрашивает разрешение на:

🔔 Показывать оповещения

Предоставлено SendPulse

Блокировать

Разрешить



[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.](#)

http://av-assembler.ru запрашивает разрешение на:

🔔 Показывать оповещения

Предоставлено SendPulse

Блокировать

Разрешить



**Зарабатывай
играя**



**Робот, который
приносит 30%
прибыли в месяц
на автомате.
Начните**



**Известный крипто
робот для
торговли. 30%
прибыли в месяц.
Бесплатно!**

Инструкция LOOP




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.

[Нужно...](#)

<http://av-assembler.ru> запрашивает разрешение на:

 Показывать оповещения

Предоставлено SendPulse

Блокировать

Разрешить

Инструкция LOOP



Инструкция LOOP в Ассемблере уменьшает значение в регистре CX в реальном режиме или ECX в защищённом. Если после этого значение в CX не равно нулю, то команда LOOP выполняет переход на **МЕТКУ**. Синтаксис:

LOOP **МЕТКА**

Состояние [флагов](#) не изменяется.

МЕТКА - это допустимый в Ассемблере идентификатор. О метках в Ассемблере я рассказывал [здесь](#).

Алгоритм работы команды LOOP:

- $CX = CX - 1$
- Если CX не равен 0, то выполнить [переход](#) (продолжить цикл)
- Иначе не выполнять переход (прервать цикл и продолжить выполнение программы)

То есть команда LOOP выполняется в два этапа. Сначала из [регистра](#) CX вычитается единица и его значение сравнивается с нулём. Если регистр не равен нулю, то выполняется переход к указанной **МЕТКЕ**. Иначе переход не выполняется и управление передаётся команде, которая следует сразу после команды LOOP.

Как выполнить цикл в Ассемблере

Выполнение цикла в Ассемблере можно организовать с помощью нескольких команд. Одна из таких команд - это команда LOOP. Команда цикла в Ассемблере всегда уменьшает значение счётчика на единицу. Это значение находится в регистре CX (или ECX). Отличия между командами цикла заключаются в следующем.

Команда LOOP
нулю. Что
число итер
затем напи
команд зап

http://av-assembler.ru запрашивает разрешение на:

🔔 Показывать оповещения

Предоставлено SendPulse

Блокировать

Разрешить

авно
исать
метку, а
к

Более понятно это будет в примере программы (см. ниже).

ПРИМЕЧАНИЕ

В качестве счётчика команда LOOP использует регистр CX в реальном режиме, и регистр ECX в защищённом режиме. Это не всегда удобно, если программу (или её часть) планируется использовать в обоих режимах. Поэтому в [системе команд процессоров Интел](#) предусмотрены две специальные команды - LOOPD и LOOPW, которые независимо от режима работы процессора в качестве счётчика используют регистры ECX и CX соответственно.

Пример программы:

```
.model      tiny
.code
ORG 100h

start:
    MOV CX, 26      ; Цикл будет выполнен 26 раз
    MOV DL, 'A'     ; CL = 41h (ASCII-код) - первая буква
    MOV BX, 676h    ; Позиция первой буквы на экране
    MOV AX, 0B800h  ; Установить AX = B800h (память VGA)
    MOV DS, AX      ; Копировать значение из AX в DS
    MOV DH, 01001110b ; CH = атрибуты цвета

abcde:
    MOV [BX], DX    ; Записать символ в видеопамять
    INC DL          ; Увеличить ASCII-код (для следующего символа)
    ADD BX, 2       ; Сместить координаты
    LOOP abcde      ; Повторить цикл

END start
```

Возможные ошибки

Начинающие довольно часто совершают одни и те же ошибки при организации циклов в Ассемблере. А именно - неправильно задают или обнуляют значение счётчика перед выполнением цикла.

При обнулении счётчика перед циклом при первой итерации цикла значение в регистре CX будет равно FFFFh (потому что команда LOOP отнимет от CX единицу, а в CX у нас был 0), и цикл в программе будет выполняться, соответственно 65536 раз.

Ещё один момент: диапазон адресов для передачи управления в команде LOOP ограничен в пределах -128...+127 байтов относительно адреса следующей команды. Если учесть, что в реальном режиме процессора средняя длина машинной команды равна 3 байта, то получается, что блок команд, выполняющихся в цикле, может состоять примерно из 42 команд. Если же в вашем цикле будет больше команд, то, например, MASM, выдаст сообщение об ошибке, которое будет выглядеть примерно так:

```
error A2075: jump destination too far : by 10 byte(s)
```

Здесь говорится, что местоположение перехода слишком далеко (примерно на 10 байт больше допустимого).

Ещё одна ошибка - это изменение значения регистра CX в теле [цикла](#). В итоге команда LOOP будет работать неправильно. К тому же при этом можно попасть в бесконечный цикл. Пример:

```
MOV CX,
top:
    INC CX
    LOOP top
```

Здесь в теле
следовател

<http://av-assembler.ru> запрашивает разрешение на:

 Показывать оповещения

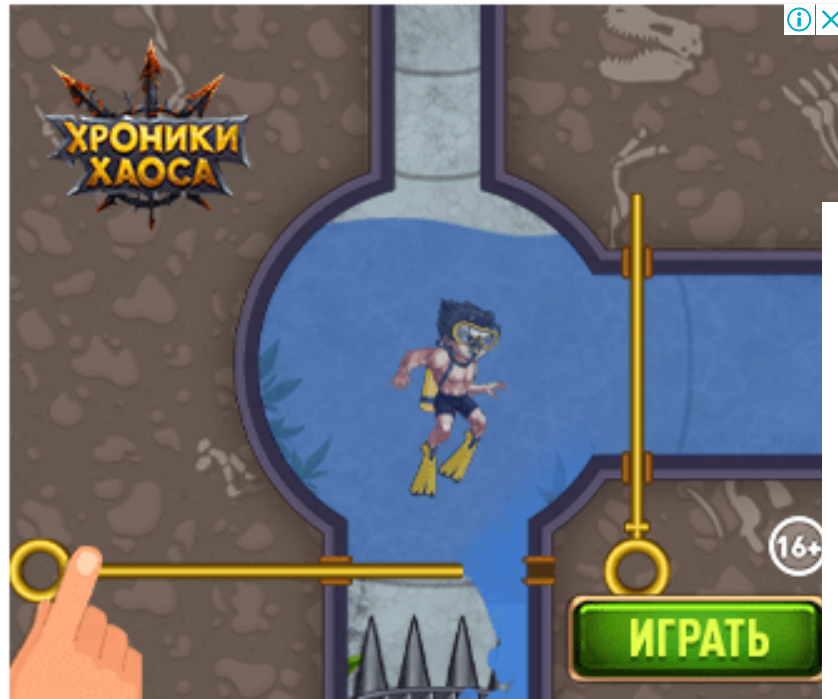
Предоставлено SendPulse

Блокировать

Разрешить

лю, и,

А теперь о происхождении мнемоники **LOOP**. В общем то это не мнемоника, а слово. В переводе с [английского](#) оно означает “петля”, “виток”, “цикл”.



[Подписаться на канал в YouTube](#)

[Вступить в группу "Основы программирования"](#)

[Подписаться на рассылки по программированию](#)



Первые шаги в программирование

Главный вопрос начинающего программиста – с чего начать? Вроде бы есть желание, но иногда «не знаешь, как начать думать, чтобы до такого додуматься». У человека, который никогда не имел дело с информационными технологиями, даже простые вопросы могут вызвать большие трудности и отнять много времени на решение. [Подробнее...](#)

Инфо-МАСТЕР®

Все права защищены ©

e-mail: mail@info-master.su

[Главная](#)

[Карта](#)

[Контакты](#)



http://av-assembler.ru запрашивает разрешение на:

🔔 Показывать оповещения

Предоставлено SendPulse

Блокировать

Разрешить