



## Как стать программистом

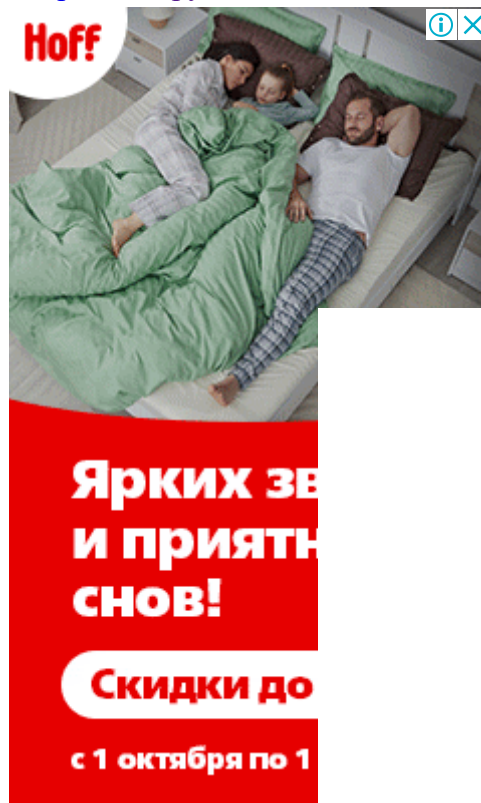
Бесплатная книга о программировании  
для начинающих и бывалых.

[Получить >>>](#)

Подписаться:



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



Микрокон  
[Изучать Б](#)

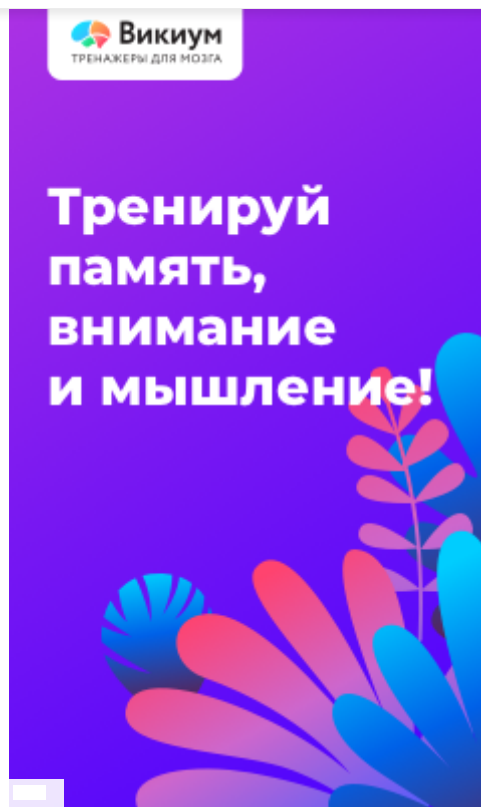
http://av-assembler.ru запрашивает разрешение на:

🔔 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**



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



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



30% прибыли в месяц на автомате. Робот для торговли криптовалютой. Для

## Команда JMP



### 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

## Команда JMP



Команда JMP - это команда безусловного перехода в Ассемблере. Выполняет, соответственно, безусловный переход в указанное место.

Синтаксис команды JMP такой:

JMP МЕТКА

МЕТКОЙ может быть один из следующих:

- Идентификатор метки в исходном коде
- Адрес в формате СЕГМЕНТ:СМЕЩЕНИЕ

Безусловный переход в Ассемблере выполняется всегда и в любом случае, если в исходном коде имеется инструкция JMP.

Никакие [флаги](#) при выполнении этой инструкции не изменяются.

## Что такое безусловный переход

Как следует из названия, это переход **без условий**. Это значит, что если в исходном коде встречается команда JMP, то эта команда выполняет переход на указанный в МЕТКЕ адрес без каких либо условий - в любом случае.

## Что такое метка в Ассемблере

Теперь нем

Метка - эт  
составлен

Пример ме

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

Показывать оповещения

Предоставлено SendPulse

Блокировать

Разрешить

ия

--- Здесь инструкции Ассемблера ---

**MyLabel:** ; Это МЕТКА

--- И здесь инструкции Ассемблера ---

Для чего нужны метки?

Для того, чтобы управлять ходом выполнения программы.

Например, в зависимости от результата выполнения какой-то команды, вам требуется направить ход выполнения программы по одному из двух путей. То есть в зависимости от результата выполнить один из двух участков кода.

Тогда каждый участок кода обозначается своей меткой. И таким образом вы будете иметь возможность пропустить какой-то участок кода и сразу перейти к другому участку.

Более подробно об этом мы будем говорить при изучении инструкций условного перехода. Эта же статья посвящена безусловному переходу, то есть когда надо просто перейти к нужному участку кода без каких либо условий.

В языках высокого уровня подобные действия называются [ветвлением](#), и реализуются с помощью соответствующих конструкций языка (таких как if...then...else или case в Паскале).

## Пример безусловного перехода

Сначала давайте подумаем, зачем нужен безусловный переход.

**Логичный вопрос:** если мы всегда и во всех случаях пропускаем участок кода, то зачем нам тогда вообще этот участок?

Вопрос закономерный, но только для новичка. На самом деле бывают разные ситуации, когда этот переход нужен.

**Пример первый:** не забывайте, что вы можете переходить по программе не только вниз, но и вверх. Так что при первом проходе участок кода можно пропустить, но, возможно, его придётся выполнить в зависимости от условий, которые появятся дальше в программе. Как-нибудь так:

```
JMP Label_2
Label_1:
--- Участок кода 1 ---
Label_2: ; Это МЕТКА
--- Участок кода 2 ---
JMP Label_1
```

Здесь при первом проходе программы мы пропускаем участок кода 1 и сразу переходим к метке Label\_2 (то есть ко второму участку). Но во втором участке мы возвращаемся к участку 1 (к метке Label\_1) и выполняем его.

**Пример второй:** может быть так, что какой-то участок кода требуется только для отладки. Тогда можно сделать так:

```
; JMP Label_2
--- Участок кода ТОЛЬКО для отладки ---
Label_2:
--- Участок кода ---
```

Обратите внимание, что этот код отлаживается только в режиме отладки.

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

🔔 Показывать оповещения

Предоставлено SendPulse

Блокировать

Разрешить

только вы

Когда отладка программы закончена, мы убираем точку с запятой перед инструкцией JMP и отладочный код выполняться не будет, так как будет выполняться безусловный переход на метку Label\_2.

А почему бы просто не убрать отладочный код?

Можно, конечно, и убрать. Но вы уверены, что он вам больше не пригодится?

В конце как обычно расскажу, почему эта команда ассемблера называется **JMP**. Это сокращение от английского слова **JUMP**, которое можно перевести как “прыжок, переход”.



[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@av-assembler.ru](mailto:mail@av-assembler.ru)

[Главная](#)

[Карта](#)

[Контакты](#)

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

🔔 Показывать оповещения


Предоставлено SendPulse

Блокировать

Разрешить



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

 Показывать оповещения

Предоставлено SendPulse

**Блокировать**

**Разрешить**