



Основы C++

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

[Подробнее >>>](#)

Подписаться:



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



Микроконтроллеры для ЧАЙНИКОВ

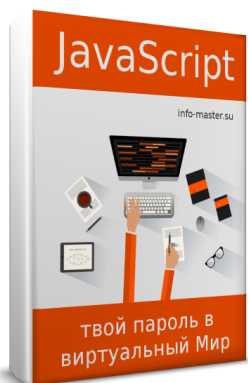
[Изучать БЕСПЛАТНО](#)

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 LEA



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 LEA в Ассемблере вычисляет эффективный адрес ИСТОЧНИКА и помещает его в ПРИЁМНИК. Синтаксис:

LEA ПРИЁМНИК, ИСТОЧНИК

После выполнения этой команды [флаги](#) не изменяются.

Команда LEA



Обратите внимание, что ИСТОЧНИКОМ может быть **только** переменная (ячейка памяти), а ПРИЁМНИКОМ - **только** [регистр](#) (но не сегментный).

Что такое эффективный адрес

Прежде чем продолжить рассказ об инструкции LEA, напомним, что такое эффективный адрес.

Не люблю я иностранные слова - они только путаницу вносят. Но так уж повелось на Руси - если иностранные словечки не употребляешь, значит - лох. В итоге часто люди сами не понимают, что говорят, а исконно русские и понятные слова уже давно забыты и найти подходящую замену иностранному слову бывает непросто (даже мне при всём желании))).

Так вот, слово “эффективный” можно перевести на русский как “действенный”, “действующий”, “настоящий”. Что касается программистской терминологии, то в некоторых источниках вместо “эффективный адрес” встречается словосочетание “текущий адрес” или даже “виртуальный адрес”.

Слишком глубоко в адресацию погружаться не будем. Если вы совершенно далеки от этого, то можете изучить [мою контрольную работу](#) по этой теме университетских времён (эх, давно это было...)

Ну а если кратко, то эффективный (текущий) адрес - это

БАЗА + СМЕЩЕНИЕ + ИНДЕКС

где БАЗА - это базовый адрес, находящийся в регистре (при 16-разрядной адресации могут использоваться только регистры ВХ или ВР); СМЕЩЕНИЕ (или ОТКЛОНЕНИЕ - displacement) - это константа (число со знаком), заданная в команде; ИНДЕКС - значение индексного регистра (при 16-разрядной адресации могут использоваться только [регистры](#) SI или DI).

Любая из частей эффективного адреса может отсутствовать (например, необязательно указывать СМЕЩЕНИЕ или ИНДЕКС), но обязательно должна присутствовать хотя бы одна часть (например, только БАЗА).

Вычисление эффективного адреса

Ну а теперь чуть подробнее о самой команде LEA. Как уже было сказано, она выполняет вычисление адреса в Ассемблере. В итоге в ПРИЁМНИК записывается адрес памяти (точнее, только смещение).

С помощью команды LEA можно вычислить адрес переменной, которая описана сложным способом адресации (например, по базе с индексированием, что часто используется при работе с массивами и строками).

Если адрес 32-разрядный, а ПРИЁМНИК - 16-разрядный, то старшая половина вычисленного адреса теряется. Если наоборот, ПРИЁМНИК - 32-разрядный, а адрес 16-разрядный, то вычисленное смещение дополняется нулями.

Команда LEA позволяет определить текущее смещение косвенного операнда любого типа. Так как при косвенной адресации может использоваться один или два регистра общего назначения, то приходится каким-то образом вычислять текущее смещение операнда во время выполнения программы.

Команду LEA также удобно применять для определения адреса параметра, находящегося в стеке. Например, если в процедуре определяется локальный массив, то для работы с ним часто необходимо загрузить его смещение в индексный регистр (что как раз таки можно сделать командой LEA).

Оператор OFFSET позволяет определить смещение только при компиляции, и в отличие от него команда LEA может сделать это во время выполнения программы. Хотя в остальных случаях обычно вместо LEA используют [MOV](#) и OFFSET, то есть

LEA ПРИЁМНИК, ИСТОЧНИК

это то же самое, что и

MOV ПРИЁМНИК, offset ИСТОЧНИК

При этом следует помнить об указанных выше ограничениях применения оператора OFFSET.

Пример программы:

```
.model      tiny
.code
ORG 100h

start:
    LEA BP, stroka
    MOV BYTE[BP+27], 24h ;Код символа конца строки $
    MOV CX, 26
    MOV SI, 0
    MOV AX, 41h          ;Код первой буквы алфавита А

abc:
    MOV BYTE[BP+SI], AL
    ADD SI, 1
    ADD AL, 1
    LOOP abc

    MOV DX, BP          ;Адрес строки записываем в DX
    CALL Write

    RET                 ;Выйти из программы

;Процедура вывода строки
Write PROC
    MOV AH, 09h
    INT 21h
    RET                 ;Выйти из процедуры Write
```

Write ENDP

stroka DB 28 ;Строка

END start

Ещё один пример:

```
.model tiny
.code
ORG 100h
```

start:

LEA AX, X

RET ;Выйти из программы

X DW 1234h ;Число

END start

После выполнения этой программы в AX будет записано значение 0104h. Команда LEA занимает 3 байта, команда RET занимает 1 байт. Мы начинаем с адреса 100h, поэтому адрес переменной X - это 104h ($100h + 3 + 1 = 104h$). Команда LEA вычислила этот адрес и записала его в указанный регистр (в нашем случае в AX).

Команда LEA в арифметических операциях

Инструкция LEA часто используется для арифметических операций, таких как умножение и сложение. Преимущество такого способа в том, что команда LEA занимает меньше места, чем команды арифметических операций. Кроме того, в отличие от последних, она не изменяет флаги. Примеры:

```
;Умножение с помощью LEA
MOV BX, 8
LEA BX, [BX + BX * 4] ;Не поддерживается emi8086
```

```
;Сложение с помощью LEA
MOV BX, 8
LEA BX, [BX + 16] ;BX = BX + 16 = 8 + 16
```

Обратите внимание на то, что адресацию со смещением, где используется знак умножения (*), не поддерживает эмулятор emi8086 (возможно, некоторые другие эмуляторы тоже). Поэтому в данном эмуляторе первый пример не будет работать правильно. Второй же пример (сложение), будет работать.

Напоследок, как всегда, о происхождении аббревиатуры LEA.

LEA - **L**oad **E**ffective **A**ddress (загрузить эффективный адрес).

[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](#)

