Учебный курс. Часть 8. Числа со знаком и без

Автор: xrnd | Рубрика: Учебный курс | 27-03-2010 | Распечатать запись

Числа со знаком и дополнительный код

Помимо того, что процессор работает с двоичными числами, эти числа могут быть со знаком или без знака. Если число без знака, то оно просто представляет собой результат перевода десятичного числа в двоичный вид. Все биты в таком числе являются *информационными* и оно может принимать только неотрицательные значения.

Для представления чисел со знаком используется специальное кодирование. Старший бит в этом случае обозначает знак числа. Если знаковый бит равен нулю, то число положительное, иначе — отрицательное. Понятно, что положительное число со знаком будет выглядеть точно так же, как и число без знака.

С отрицательными числами чуть сложнее. Исторически для представления отрицательных чисел в компьютерах использовались разные виды кодирования: прямой, обратный и дополнительный код. В настоящее время наиболее часто используется дополнительный код, в том числе и в процессорах x86.

Чтобы сделать из положительного числа отрицательное, необходимо проинвертировать все его биты (0 заменяем на 1, а 1 заменяем на 0) и затем к младшему разряду прибавить единицу. Например, представим -5 в дополнительном коде:

В обратную сторону переводится точно также 🙂

Синтаксис FASM

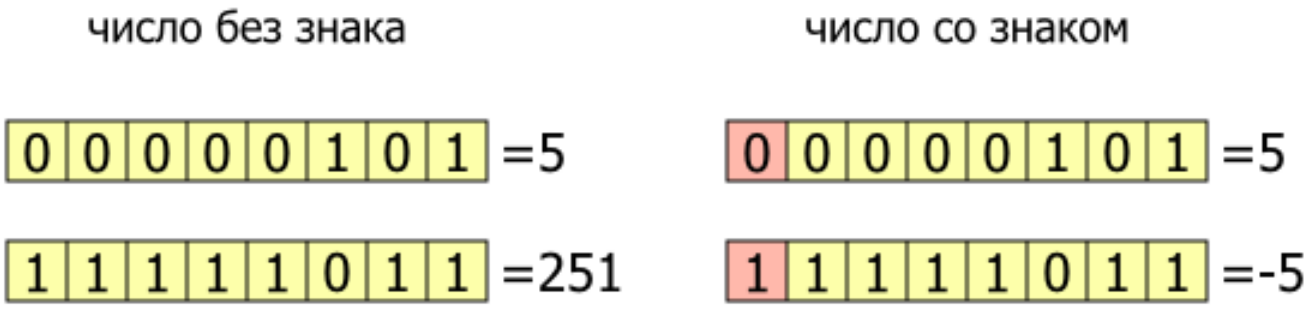
Для записи отрицательного числа в программе на ассемблере используется символ ‘-‘, например:

```
x db -5
```

Кстати, это работает и с числами в других системах счисления, и даже с символами 🙂

```
y db -25h
z db -77o
k db -101b
s db -'a'
```

Со знаковыми и беззнаковыми числами нужно быть внимательным, потому что только вы знаете, какие числа используются в вашей программе! Процессору абсолютно по барабану, какие данные он обрабатывает, поэтому невнимательность может привести к ошибке. Один и тот же байт может интерпретироваться по-разному, в зависимости от того со знаком число или без. Например, числу со знаком -5 соответствует число без знака 251:



Диапазоны значений чисел со знаком и без

При программировании на ассемблере (как, впрочем, и на многих других языках) необходимо учитывать ещё один важный момент. А именно — ограничение диапазона представления чисел. Например, если размер беззнаковой переменной равен 1 байт, то она может принимать всего 256 различных значений. Это означает, что мы не сможем представить с её помощью число, больше 255 (11111111_2). Для такой же переменной со знаком максимальным значением будет 127 (01111111_2), а минимальным -128 (10000000_2). Аналогично определяется диапазон для 2- и 4-байтных переменных.

Кстати, так как процессор Intel 8086 был 16-битным и обрабатывал за одну команду 16-бит, то 16-битная переменная называется *слово (word)*, а 32-битная — *двойное слово (double word, dword)*. Эти названия сохранились в ассемблере даже для 32-битных процессоров (и в WIN32 API, например). И от них же происходят названия директив *dw (Define Word)* и *dd (Define Dword)*. Ну а *db* — это *Define Byte*.

Для наглядности вот табличка диапазонов чисел:

Размер переменной	Число без знака		Число со знаком	
	min	**max**	**min**	**max**
байт	00000000	11111111	10000000	01111111
	0	255	-128	127
слово	00000000 00000000	11111111 11111111	10000000 00000000	01111111 11111111
	0	65 535	-32 768	32 767
двойное слово	0000…0000	1111…1111	1000…0000	0111…1111
	0	4 294 967 295	-2 147 483 648	2 147 483 647
и т.д.	…	…	…	…

Если результат какой-то операции выйдет за пределы диапазона представления чисел, то случится *переполнение* и результат будет некорректным. (Например, при сложении двух положительных чисел, можно получить отрицательное число!) Поэтому нужно быть внимательным при программировании и предусмотреть обработку таких ситуаций, если они могут возникнуть.

Следующая часть »

Комментарии:

ratiug
09-11-2010 10:38

Спасибо за этот учебный курс, читаю с самого начала, занимался ассемблером лет 6 назад, забылось многое, сейчас вот читаю и вспоминаю =) очень понятно изложен материал. С числами со знаками не отложилось и в прошлый раз..или просто не предал этому значения, но в процессе отладки встречаются знаковые и беззнаковые операторы перехода, и тут приходиться анализировать аргументы. Материал помог разложить мне у себя в мозгу всё по полочкам =) ..Спасибо!!!

[Ответить]

xrnd
10-11-2010 17:31

Спасибо. Такие комментарии мне нравятся 🙂

[Ответить]

oleg
15-11-2010 00:04

2ratiug

Только не операторы перехода, а операнды сравнения.

[Ответить]

xrnd
15-11-2010 17:10

Если быть совсем точным, то команды условного перехода. В ассемблере знаковые и беззнаковые операнды никак не проверяются компилятором и сравниваются одной и той же командой CMP. Разница только в выборе команды условного перехода.

[Ответить]

anton
16-02-2011 01:00

Приветствую, xrnd и всех. Такой вот вопрос замучил: почему при сложении 106+25=131 возникает переполнение, а в случае 2+129=131 нет ? Складываю в байтовом регистре, т.е. в обоих случаях получаю одно и то же отрицат. число.

[Ответить]

xrnd
18-02-2011 13:21

Тут надо быть внимательным с флагами.
Дело в том, что числа со знаком и числа без знака складываются и вычитаются совершенно одинаково. Разница только в интерпретации флагов и знакового бита.

Флаг переполнения OF имеет смысл только для чисел со знаком.
Переполнение возникает в следующих случаях:
1. положительное + положительное = отрицательное
2. отрицательное + отрицательное = положительное

В этих случаях результат сложения некорректен. Например, два положительных числа должны давать в сумме положительное, а не отрицательное.

Если записать твои примеры в виде чисел со знаком, то всё становится понятным:
106+25 = -125 (переполнение!)
2 + (-127) = -125 (нет переполнения)

Для чисел без знака проверять переполнение нужно по флагу переноса CF. Если был перенос из старшего разряда, то результат вышел за допустимые пределы.

[Ответить]

anton
18-02-2011 16:16

Значит, если подразумевается, что числа со знаком, поднятый of будет означать неверный результат, если же числа беззнаковые, на of можно не обращать внимания, а смотреть на cf. Спасибо, теперь понятно.

[Ответить]

Денис
11-11-2011 16:16

я что то не понял при записи 101b=5 но при 010b получается 2. Так как при инвертации нельзя записать число с начальным нулем получается 10b
А при -5 получаеться 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1011b
я что то не понимаю вас

[Ответить]

Sanya
03-05-2015 18:30

Имеются два числа: 11111011 и 11111011. Как ассемеблер понимает что нужно выводить отрицательное число или положительное?

[Ответить]

Sammy
27-03-2018 11:28

Мне тоже интересно. Вопрос еще актуален, ответьте пожалуйста, о, гуру ассемблера 🙂

[Ответить]

Arct
16-06-2019 15:37

Ассемблер — никак… Это задача программиста на ассемблере…
А вот отдельные команды, IDIV (целочисленное деление со знаком) например, по наличию «1» в старшем (знаковом) разряде своих операндов.

[Ответить]

Ваш комментарий

Имя *

Почта (скрыта) *

Сайт

Добавить

Уведомить меня о новых комментариях по email.

Уведомлять меня о новых записях почтой.