Алгоритм ввода целых чисел

Состоит из двух этапов:

- Ввод строки символов в буфер с клавиатуры, в результате чего получиться **символьное** представление числа в ASCII коде;
- Преобразование строки символов в коде ASCII во внутреннее представление числа.

Алгоритм преобразования строки символов во внутреннее представление целого числа будет обратным алгоритму вывода целого числа.

Рассмотрим алгоритм на примере.

Алгоритм получения внутреннего представления числа 359.

После ввода числа 359 в информационной части буфера клавиатуры сформируется строка рис.2.

Символьное представление	‘ 3’		‘5‘		‘9’	
Нех-код	3	3	3	5	3	9
Полубайты	Старший	Младший	Ст.	Млад.	Ст.	Млад.

Рис.2. Содержимое информационной части буфера клавиатуры при вводе числа 359.

Заметим, что содержимое младших полубайт - это и есть нужные нам составляющие числа. Вспомним, что десятичная система счисления является позиционной, значит наше число может быть представлено $359=3*10^2+5*10^1+9*10^0$ на этих идеях базируется основной алгоритм преобразования строки символов во внутреннее представление целого положительного числа.

1. Выделить старшую цифру числа (младшие четыре бита-младший полубайт) и занести в промежуточный результат. Промежуточный результат = 3.
2. Выделить следующую цифру числа. Если у числа цифр больше нет, перейти к пункту 6. Следующая цифра числа = 5.
3. Умножить промежуточный результат на число 10. 3*10=30.
4. Добавить выделенную в пункте 2 цифру к произведению и результат занести в промежуточный результат. Промежуточный результат=5+30=35.
5. Перейти к пункту 2.
   Начинается вторая итерация нашего алгоритма:
   2-2) следующая цифра числа 9.
   3-2) 35*10=350
   4-2) промежуточный результат = 9+350=359.
   5-2) перейти к пункту 2.
   Начинается третья итерация нашего алгоритма:
   2-3) следующая цифра числа отсутствует. Переходим к пункту 6.

6. Конец работы алгоритма. Промежуточный результат - это и есть внутреннее представление нашего числа.

Вывод целых чисел

Вывод целых чисел состоит из двух этапов:

- Преобразование внутреннего представления числа в строку символов (код ASCII);
- Вывод полученной строки символов.

Если внимательно посмотреть на символьное представление цифр **‘0’-‘9**’ в коде ASCII по можно заметить, что они в шестнадцатеричном виде имеют коды 30h-39h, на этом и построен алгоритм преобразования.

Алгоритм преобразования целых (16-разрядных) чисел со знаком сводиться к последовательному делению числа (в его внутреннем, машинном представлении) на константу 10 (поскольку нас интересует десятичное представление числа).

В процессе целочисленного деления получаются остатки, которые преобразуются в символьный вид и заносится в буфер вывода.

1. Исходное целое число [-32768, 32767] поместить в регистр ax;
2. Очистить буфер вывода (например, занести символ 0) для последующего размещения в нем строкового представления числа: ЦЦЦЦЦЦ или – ЦЦЦЦЦ (которые будут представляться цифрами, где Ц – числа от 0 до 9). Таким образом, минимальная длинна буфера составляет 6 символов (buff db 6 dup(‘_’));
3. Все последующие арифметические операции производятся с положительным числом. Поэтому нужно проверить знак числа.
4. Если число отрицательное, то сделать его положительным.
5. Выполнить целочисленное беззнаковое деление числа на 10.
6. Остаток от целочисленного деления перевести в символьное представление. Для этого достаточно остаток сложить с символом 0 (30h).
7. Занести символьное представление остатка в буфер вывода (остатки будут заноситься в буфер, начиная с конца буфера).
8. Проверить частное. Если оно не равно 0 перейти на пункт 5.
9. Если исходное число было отрицательным, то занести в буфер вывода символ ‘-‘ .
10. Конец работы алгоритма.

Поясним этот алгоритм (п. 3-9) на примере:

Пусть нужно преобразовать строку число <AX>=-157.

3) Проверить знак числа -157:
Используем команду
OR[1] AX, AX

4) В примере число отрицательное, сделаем его положительным, используя команду:

---

[1] **and|or** приёмник, источник – выполняет логическое побитовое И|ИЛИ над приёмником и источником, и помещает результат в приёмник. Часто используется для выборочного обнуления|объединичивания отдельных битов. Например, команда and al, 00001111b обнулит старшие 4 бита регистра al, а младшие не изменит

NEG[2] AX

5) Делим наше положительное число (157) на 10 (константа 10 хранится в регистре SI)
МЕТКА1:
SUB DX, DX
DIV SI
Получим частное в <AX>= 15, остаток <DX> = 7

6) Переводим остаток от целочисленного деления (7) в символьное представление:
ADD DX, '0'
Получим <DX>=37h

7) Заносим в символьное представление остатка в буфер вывода:
DEC[3] BX
MOV Byte PTR [BX], DL

8) Проверим частное на равенство его нулю:
OR AX, AX
JNZ МЕТКА1

Наше частное <ax> =15, не равно 0 . Поэтому переходим опять на пункт 5 и опять повторяем все пункты. В результате получим следующее:
Частное в <AX> = 1 , остаток <DX> = 5 . Преобразуем остаток в символ, получим <dx>=35h и опять заносим его (символ 5) в буфер вывода. Теперь наше новое частное <AX> = 1 , тоже не равная 0, поэтому переходим опять на п.5. и повторяем все пункты. В результате получаем следующее: частное <AX> = 0, остаток <DX>=1. Преобразуем остаток в символ, получим <DX>=31h и опять заносим его ('1') в буфер вывода рис.1.
Теперь мы получим частное <AX> = 0, поэтому процесс последовательного деления на 10 прекращаем.
9) Поскольку наше исходное число было отрицательным, то в буфер вывода нужно занести его знак минус:
DEC BX
MOV Byte PTR [BX], '-'
10) На этом алгоритм заканчивается.

Символьное представление	' '	' '	'-'	'1'	'5'	'7'	'$'
Hex-код	20	20	2d	31	35	37	24

Рис.1. Внутреннее представление буфера вывода для числа -157.

[2] NEG (Получение противоположного значения). Меняет знак операнда на противоположный.
[3] DEC (Декремент). Уменьшает содержимое операнда на единицу.