Вычитание - это обратная операция к суммированию. Поэтому совершенно естественно желание использовать в реализации вычитателя те же приемы что и в сумматоре, либо по возможности задействовать в выполнении обеих операций один механизм.

Когда мы изучали арифмометры, то могли наблюдать два варианта:
1. вычитание - это то же суммирование, просто надо крутить ручку в обратном направлении.
2. вычитание делается суммированием с дополнением (дополнительным кодом) вычитаемого (Паскалина).

Сейчас же, когда сумматор состоит из электронных компонентов, прокрутить ручку назад не получится, нужен параллельный механизм. А вот суммирование с дополнением нас вполне устраивает, необходимо только разобраться что это за дополнение такое и как его получить. Отметим, что электронный сумматор работает точно как механический в Паскалине - в каждом разряде увеличивает цифру, при переходе через 0 - добавляет единицу в старший разряд. Система счисления другая - 2 вместо 10, но, как мы увидим, это не важно.

Пусть сумматор имеет ограничение в три десятичные цифры (три колеса с цифрами). В этом случае 999 + 1 (+1 соответствует вращению на одно деление младшего колеса) приведет к переполнению и если ничего не предпринять, число обратится в 0. Так и должно быть, если мы работаем с беззнаковыми числами. А что в случае отрицательных чисел ?

Что, если мы станем хранить знак в виде дополнительного четвёртого колеса, на котором вместо цифр чередуются знаки +-+-... ?
Первая проблема, которая бросается в глаза - теперь у нас два ноля, +000 и -000!
Вторая - если к -001 прибавить 1, получится -2. Это неприемлемо.

Данные проблемы возникают из-за того, что числа одновременно выступают в двух ролях: во первых - это собственно значение числа, во-вторых - приемлемый для сумматора способ хранения числа. Для отрицательных чисел роли разошлись - сумматор вместо суммирования начал отнимать. Надо договориться о таком способе хранения отрицательных чисел, с которыми он умеет работать.

Если к числу 999 добавить 1, произойдет переполнение и мы увидим на колёсах -000. Было бы логично, если бы код -000 означал самое маленькое отрицательное число. Наоборот, если бы мы могли от 000 отнять 1 (прокрутить младшее колесо назад), получили бы -999, логично обозначить этим кодом число -1.

Теперь противоречие исправлено - если к самому маленькому числу (код -000) добавлять по единице, рано или поздно придём к -1 (код -999). Если добавить еще 1, получим 0, как и ожидали. Круг замкнулся.

Осталось понять что это за самое маленькое число. Чтобы получить -1 (хранится как

-999) из -000 и к нему надо прибавить 999. Или от -1 отнять 999, получим -1000. Вот наше самое маленькое число. Кажется странным, что оно не соответствует самому большому положительному 999. Это плата за отсутствие значение -0. Мы сдвинули весь диапазон отрицательных чисел вперёд на единицу, в результате отрицательных чисел стало больше чем положительных.

Итак, мы придумали способ хранения отрицательных чисел. Это и называется “дополнительным кодом”. Дополнительный код - это то, как придётся хранить число, если его умножить на -1. Но как получить дополнительный код - вот берём число 123, какой у него дополнительный код?

1) Надо каждую цифру заменить на ее дополнение. Дополнением цифры называется ее разность с максимальной цифрой в системе счисления.
В десятичной системе счисления
Доп(A) = 9 - A.
После этого число 123 превращается в 876.
2) Сменим знак 876 превратится в -876. Это так называемое простое дополнение.
Теперь если 123 (или любое другое число) сложить со своим простым дополнением, получим -999, это простое дополнение 0, фактически -0.
3) Отнимаем единицу, чтобы сдвинуть диапазон чисел и избавиться от значения -0.
-876 превратится в -877. Вот мы и получили расширенное дополнение. Теперь если сложить 123 (или любое другое число) со своим расширенным дополнением, получим 0. Что и требовалось.

Обычно, когда говорят просто дополнительный код или дополнение, имеют ввиду именно расширенное дополнение.

Число	Простое дополнение	Расширенное дополнение
1000	-------	-000
999	-000	-001
100	-899	-900
010	-989	-990
001	-998	-999
000	000	000
-000	-999	000

Как насчет двоичной системы счисления? Всё то же самое с учетом того, что дополнения цифр считаются не от 9, но от 1.
Приятный бонус в двоичной системе то, что одноразрядный флаг переполнения естественным образом вписывается в двоичное же представление значащих цифр.
Построим таблицу кодов для восьмиразрядных чисел.

Число	Двоичное представление	Простое дополнение	Расширенное дополнение
128	----------	----------	$1000\ 0000_2$
127	$0111\ 1111_2$	$1000\ 0000_2$	$1000\ 0001_2$
80	$0101\ 0000_2$	$1010\ 1111_2$	$1011\ 0000_2$
20	$0001\ 0100_2$	$1110\ 1011_2$	$1110\ 1100_2$
2	$0000\ 0010_2$	$1111\ 1101_2$	$1111\ 1110_2$
1	$0000\ 0001_2$	$1111\ 1110_2$	$1111\ 1111_2$
0	$0000\ 0000_2$	$0000\ 0000_2$	$0000\ 0000_2$

Итого: в восьми разрядах кодируются знаковые числа от -128 до +127.
Обратите внимание, как естественно и логично старший - знаковый разряд вписался в формат хранения знаковых целых чисел.
А также не пропустите тот факт, что простое дополнение числа получается инверсией разрядов. В самом деле дополнением для двоичной системы счисления будет

Для 0: 1 - 0 = 1
Для 1: 1 - 1 = 0

Иными словами, умножение целого знакового числа *val* на -1 равносильно выражению

*val * -1 = negate(val) + 1*

Проверим, умножим -20 на -1:

-20 = $1110\ 1100_2$
negate ($1110\ 1100_2$) = $0001\ 0011_2$
$0001\ 0011_2 + 1 = 0001\ 0100_2 = 2^4 + 2^2 = 16 + 4 = 20$

И, наконец, последнее, на что стоит обратить внимание, знак числа определяется по значению самого старшего разряда, для положительных чисел (включая 0) этот разряд нулевой, для отрицательных чисел, равен единице.
Собственно, поэтому он и называется - знаковый разряд.