Базовый одинарный формат

Слово длиной 4 байта, в котором используется смещенный порядок (характеристика) числа, дано на рис.

Рис. Базовый одинарный формат
с плавающей запятой

Запятые для характеристики и мантиссы фиксированы. Характеристика – целое число без знака, для нее запятая расположена после младшего разряда. Для мантиссы запятая расположена перед старшим разрядом.

$X = 128 + q$ – смещенный порядок ($2^n - 1 = 255$ – максимальное число в восьми битах). Характеристика числа всегда положительна:

$$X \geq 0,$$
$$q = 127, \ X = 255,$$
$$q = -128, \ X = 0.$$

Это упрощает выполнение арифметических операций, так как не надо проверять знак порядка. В этом формате

$$A_{\max} \approx 1 * 2^{127} \approx 10^{38},$$
$$10^x = 2^{127},$$
$$X = 127 \lg 2 = 127 * 0{,}3 = 38.$$

Поэтому диапазон чисел такой: $N = \pm 10^{\pm 38}$.

По сравнению с классическим форматом диапазон чисел значительно шире, но мантисса числа на 1 бит меньше. Поэтому точность представления должна быть хуже, но так как число всегда нормализовано, то первую единицу после запятой не хранят, а подразумевают, поэтому точность чисел такая же, как в классическом формате. В этом формате используется так называемая ***скрытая единица мантиссы***.

Базовый двойной формат

Здесь слово длиной 8 байт (.Рисуем рис.), смещение порядка составляет 1024.

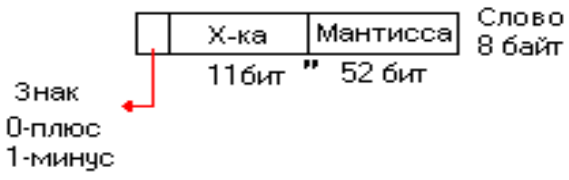


Рис. Базовый двойной формат

Характеристика $X = 1024 + q$. Порядок может находиться в пределах $-1024 \le q \le 1023$.

Диапазон чисел следующий:

$$A_{\max} = 1 * 2^{1024};$$

$$10^x = 2^{1024};$$

$$X = 1024 \lg 2 = 1024 * 0,3 = 308;$$

$$N \cong \pm 10^{\pm 308}.$$

В базовых форматах значение характеристики, равное нулю, соответствует нулевому числу, а значение характеристики, равное максимуму, соответствует бесконечности.

Мантисса длиной 24 бита соответствует точности представления числа 6 – 7 десятичных цифр. Мантисса длиной 52 бита соответствует точности представления 16 – 17 десятичных цифр.

Имеются также и расширенные форматы, но их мы не рассматриваем.

МАШИННЫЕ КОДЫ

Независимо от формы записи чисел с фиксированной или плавающей запятой все числа в ЭВМ представляются в виде специальных кодов – прямом, обратном или дополнительном.

Прямой код используется для хранения чисел в памяти и выполнения операции умножения.

Обратный и дополнительный коды используются для сложения положительных и отрицательных чисел.

Рассмотрим машинные коды на примере чисел с фиксированной запятой.

Прямой код:

$$[A]_{\text{пр}} = 0 \ \ a_n a_{n-1} \ldots a_1 a_0, \ \ A \ge 0;$$

$$[A]_{\text{пр}} = 1 \ \ a_n a_{n-1} \ldots a_1 a_0, \ \ A \le 0.$$

Знак плюс кодируют нулем, а знак минус – единицей. Знак числа обязательно должен быть в любом машинном коде.

Например,

Число	Прямой код
+ 1101	01101
– 1101	11101
+ 0,1101	01101
– 0,1101	11101
– 0,0000	10000
+ 0,0000	00000

Запятая в коде не пишется. Число нуль в прямом коде имеет двойное изображение – положительное и отрицательное.

Обратный код:

$$[A]_{\text{обр}} = 0\ \ a_n a_{n-1} \ldots a_1 a_0, \quad A \geq 0;$$
$$[A]_{\text{обр}} = 1\ \ \overline{a}_n \overline{a}_{n-1} \ldots \overline{a}_1 \overline{a}_0, \quad A \prec 0,$$

где $\overline{a_i} = 1 - a_i$ – дополнение числа до 1 (инверсия разрядов двоичного числа).

Например,

Число	Обратный код
+ 1101	01101
– 1101	10010
– 0,1101	10010
+ 0,0000	00000

Дополнительный код:

$$[A]_{\text{доп}} = 0\ \ a_n a_{n-1} \ldots a_1 a_0, \quad A \geq 0;$$
$$[A]_{\text{доп}} = 1\ \ \overline{a}_n \overline{a}_{n-1} \ldots \overline{a}_1 \overline{a}_0 \ \ + 00 \ldots 01, \quad A \prec 0,$$

где $\overline{a_i} = 1 - a_i$ – дополнение числа до 1 (инверсия разрядов двоичного числа).

Дополнительный код числа – это обратный код плюс единица в младший разряд.

Например,

Число	Дополнительный код
+1101	01101
– 1101	10011
– 1100	10100

Дополнительный код правильной дроби – это дополнение числа до основания системы счисления. $A + [A]_{\text{доп}} = 10$, где 10 – основание системы счисления.

Дополнительный код n-разрядного целого отрицательного числа есть результат вычитания этого числа из единицы с (n + 1) нулями. Так, для числа $A = -1101$ ($n = 4$) $[A]_{\text{доп}} = 100000 - 1101 = 10011$.

Для положительных чисел прямой, обратный и дополнительный коды совпадают.

ОПЕРАЦИИ НАД ЧИСЛАМИ В МАШИННЫХ КОДАХ

Операции с фиксированной запятой

Сложение чисел

В вычислительной технике благодаря машинным кодам операция вычитания заменяется операцией сложения с числом обратного знака.

$$A - B = A + (-B).$$

Сложение в обратном коде.

Пусть даны два числа A и B. Надо найти их сумму:

$$A = -0,1101;$$
$$B = +0,0010.$$

Выполним сложение в обратном коде. При этом биты знака участвуют в сложении наравне со значащими разрядами:

$$[A]_{\text{обр}} = {}_{+}10010$$
$$[B]_{\text{обр}} = \ 00010$$
$$[C]_{\text{обр}} = \ \overline{10100}$$

Ответ: $C = -0{,}1011$.

Теперь сложим два других числа:

$$A = -0{,}1101$$
$$B = -0{,}0010$$
$$[A]_{ОБР} = {}_{+}10010$$
$$[B]_{ОБР} = 11101$$
$$1 \leftarrow {}_{+}01111$$
$$\hookrightarrow 1$$

При сложении в обратном коде единица переноса из старшего (знакового) бита добавляется в младший разряд результата – имеет место так называемый циклический перенос. Результат сложения $[C]_{\text{обр}} = 10\,000$, а ответ $C = -0{,}1111$.

Сложение в дополнительном коде.

Выполним сложение тех же чисел:

$$A = -0{,}1101$$
$$B = +0{,}0010$$
$$[A]_{\text{доп}} = 10011$$
$$[B]_{\text{доп}} = 00010$$
$$[C]_{\text{доп}} = {}_{-}10101 \quad \text{дополнительный код результата}$$
$$\overset{1}{[C]_{\text{обр}}} = \overline{10100}$$

Теперь возьмем другие числа:

$$A = -0{,}1101$$
$$B = -0{,}0010$$
$$[A]_{\text{доп}} = 10011$$
$$[B]_{\text{доп}} = 11110$$
$$[C]_{\text{доп}} = 1 \leftarrow 10001$$

В дополнительном коде единица переноса из знакового бита отбрасывается. Тогда $[C]_{\text{обр}} = 10000$, а ответ равен $C = -0{,}1111$.

При сложении обязательно выравнивание разрядов слагаемых нулями (не кодов!). Для отрицательных чисел эти выравнивающие нули превращаются в единицы при инвертировании. Знаки чисел (крайние левые биты кодов) обязательно находятся один под другим.

Аналогично складывают и целые числа. Например, сложить в разрядной сетке 1 байт два числа $C = A + B$, где $A = +\,9_{10}$ и $B = -\,7_{10}$. Разместим их в фиксированной разрядной сетке – в 8 битах:

$$[A]_{\text{пр}} = 0\ 0\ 0\ 0\ 1\ 0\ 0\ 1$$
$$[B]_{\text{пр}} = 1\ 0\ 0\ 0\ 0\ 1\ 1\ 1$$

$[A]_{доп} = 0\ 0\ 0\ 0\ 1\ 0\ 0\ 1$
$[B]_{доп} = 1\ 1\ 1\ 1\ 1\ 0\ 0\ 1$
$[C]_{доп} = 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0$
$[C]_{пр} = 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0$

В результате сложения получилось положительное число, поэтому других преобразований ответ не требует.

Умножение двоичных чисел

Умножение выполняется по тем же правилам, что и десятичное умножение, т. е. перемножаются модули чисел, а знак результата получается сложением по модулю два знаков сомножителей.

Известно, что произведение двух n-разрядных чисел есть число $2n$-разрядное

$$n * n = 2n .$$

Возьмем два целых, четырехразрядных двоичных числа:

$$A = -0101;$$

$$B = +0010 \ \ (n = 4);$$

$$C = A * B.$$

Знак произведения $\text{sign}\, C = \text{sign}\, A \oplus \text{sign}\, B = 1 \oplus 0 = 1$. Перемножаем модули

$[A]_{пр} = 10101$
$[B]_{пр} = 00010$

$$\begin{array}{r} \times\ 0101 \\ \underline{\quad 0010} \\ 0000 \\ 0101 \\ 0000 \\ \underline{0000} \\ 00001010 \\ \underbrace{}_{n=4}\ \underbrace{}_{n=4} \end{array}$$

При умножении целых чисел в качестве результата берутся n младших разрядов. При этом в старших n-разрядах должны быть нули. Если это не так, то имеет место переполнение разрядной сетки машины. Для расширения разрядной сетки нужно добавить слева нули. Теперь умножим два дробных числа:

$A = -0{,}101 (n = 3)$
$B = 0{,}010$
$\text{sign}\ C = \text{sign}\ A \oplus \text{sign}\ B = 1 \oplus 0 = 1$

$$\begin{array}{r} \times\ 101 \\ \underline{\quad 010} \\ 000 \\ 101 \\ \underline{000} \\ \underbrace{0{,}001}_{n=3}\ \underbrace{010}_{n=3} \end{array} \longrightarrow C_0 = 0{,}001_2 = 1 * 2^{-3} = 0{,}125$$

При умножении дробных чисел в качестве результата выбираются n старших разрядов. При этом происходит округление по правилу: если в $(n + 1)$-разряде была единица, то к ответу добавляется единица в младший разряд, если в $(n + 1)$-разряде был ноль, то к ответу ничего не добавляется. В примере имеем ноль, поэтому ничего не добавляем. Проверим ответ в десятичной системе:

$$A = 1 * 2^{-1} + 1 * 2^{-3} = 0{,}625; \quad B = 1 * 2^{-2} = 0{,}25;$$

$$C_0 = A * B = 0{,}625 * 0{,}25 = 0{,}156;$$

$\Delta = 0{,}156 - 0{,}125 = 0{,}031$ – абсолютная ошибка;

$$\delta = \frac{\Delta}{C_0} = \frac{0{,}031}{0{,}156} = 20\% \text{ – относительная погрешность.}$$

Округление при умножении дробных чисел вносит значительную погрешность (операции же с целыми числами выполняются абсолютно точно!).

Погрешность, возникающая при умножении дробных чисел, равна

$$\Delta = \pm\frac{1}{2} * 2^{-n} = \pm 2^{-1} * 2^{-n} = \pm 2^{-(n+1)}.$$

Чтобы не потерять точность при вычислениях, нужно расширить разрядную сетку путем добавления нулей справа.

Операции с плавающей запятой

Операции над числами с плавающей запятой выполняются по тем же правилам. Мантиссы отрицательных чисел вступают в операцию в обратном или дополнительном коде. Перед сложением чисел сначала выравнивают порядки – приводят число к большему порядку и затем складывают мантиссы. Полученный результат нормализуют. Для умножения чисел перемножают мантиссы по правилам двоичной арифметики, а порядки складывают.

В качестве примера рассмотрим сложение двух действительных десятичных чисел A = 3,23 и B = – 14,85 в классическом формате с плавающей запятой, используя дополнительный код. $C = A + B = ?$

Решение начнем с перевода чисел в двоичную систему счисления:

$A = 3{,}23_{10} = +\ 11,\ 0011101_2 = +\ 0{,}110011101 * 2^{10}$;

$B = -\ 14{,}85_{10} = -\ 1110{,}1101100_2 = -\ 0{,}11101101100 * 2^{100}$.

Количество разрядов после запятой в ненормализованном двоичном числе равно $n_2 = n_{10} / 0{,}3 = 2 / 0{,}3 = 7$.

Занесем эти числа в разрядную сетку 4 байта:

$[A]_{\text{пр}}$ = 0 11001110100000000000000 0 000010;

$[B]_{\text{пр}}$ = 1 11101101100000000000000 0 000100.

Приводим операнды к большему порядку ($4_{10} = 100_2$):

$[A]_{\text{пр}}$ = 0 00110011101000000000000 0 000100.

Записываем дополнительные коды операндов:

$[B]_{\text{обр}}$ = 1 00010010011111111111111 0 000100;

$[B]_{\text{доп}}$ = 1 00010010100000000000000;

$[A]_{\text{доп}}$ = 0 00110011101000000000000;

$[C]_{\text{доп}}$ = 1 01000110001000000000000 – результат сложения;

$[C]_{\text{обр}}$ = 1 01000110000111111111111;

$[C]_{\text{пр}}$ = 1 10111001111000000000000 0 000100.

Ответ в двоичной системе счисления:

$$C = -0{,}10111001111 * 2^{100} = -1011{,}1001111_2 =$$

$$= -(2^3 + 2^1 + 2^0 + 2^{-1} + 2^{-4} + 2^{-5} + 2^{-6} + 2^{-7}) \approx -11{,}617_{10}.$$

Точный ответ: $C_0 = -14{,}85 + 3{,}23 = -11{,}62$.

Появилась погрешность за счет перевода чисел в двоичную систему счисления.

ДВОИЧНО-ДЕСЯТИЧНАЯ СИСТЕМА КОДИРОВАНИЯ

Все операции в ЭВМ выполняются в двоичной системе счисления. Однако человеку удобно вводить информацию и получать результаты вычислений в десятичной системе счисления. Для этого используются так называемые, двоично-десятичные коды. В них один десятичный разряд представляют четырьмя двоичными разрядами (тетрадой). При помощи 4 бит можно закодировать 16 различных символов (цифр). Существует много разных систем кодирования [10], но наиболее широко применяется код прямого замещения – код 8–4–2–1 (это веса двоичных разрядов влево от запятой). Составим таблицу соответствия двоично-десятичного кода и десятичных цифр (табл. 2).

Т а б л и ц а 2

Соответствие двоично-десятичного кода и десятичных цифр

Двоично-десятичный код				Десятичная цифра
8	4	2	1	
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9

Остальные комбинации двоичного кода являются лишними (запрещенными). Запишем пример двоично-десятичного кода:

1258 = 0001 0010 0101 1000;
589 = 0000 0101 1000 1001.

Сложение двоично-десятичных чисел

Сложение двоично-десятичных чисел производится по правилам двоичной арифметики, с учетом переносов. Пусть имеем два десятичных числа A и B. Требуется найти сумму $C = A + B$.

В каждой тетраде выполняется сложение трех чисел – двух слагаемых и переноса из предыдущего разряда, т.е. $a_n + b_n + p_{n-1}$. При этом возможны такие ситуации.

1. $a_n + b_n + p_{n-1} < 10$; $A = 14$; $B = 23$; $C = A + B = 37$.

A = 0001 0100;
B = 0010 0011;
C = 0011 0111 $\Rightarrow$ 37.
Ответ получился верный.

2. $a_n + b_n + p_{n-1} > 15$; $A = 47$; $B = 39$; $C = A + B = 86$.
A = 0100 0111;
B = 0011 1001;
C = 1000 $\leftarrow$0000 $\Rightarrow$ 80.
Ответ получился неверный, так как был перенос из младшей тетрады в старшую. Тетрада переполняется числом 16, т.е. единица межтетрадного переноса уносит в старшую тетраду 16, а не 10 единиц, как в десятичной системе счисления (шесть лишних единиц!). Поэтому результат необходимо скорректировать путем добавки + 6. Выполним коррекцию:
C = 1000 0000
0000 0110 – коррекция (+ 6).
Ответ 1000 0110 $\Rightarrow$ 86.
Ответ правильный.
3. $10 \le a_n + b_n + p_{n-1} \le 15$; $A = 47$; $B = 36$; $C = A + B = 83$.
A = 0100 0111;
B = 0011 0110;
C = 0111 1101 $\Rightarrow$ Ответ неверный, хотя и нет межтетрадного переноса, но имеется запрещенная комбинация. Необходимо вызвать искусственное переполнение тетрады путем добавки + 6. Выполним коррекцию:
C = 0111 1101
0000 0110 – коррекция (+ 6).
Ответ 1000 $\leftarrow$ 0011 $\Rightarrow$ 83.
Теперь ответ правильный.

Внимание! Коррекция результата выполняется только один раз, поэтому межтетрадный перенос при коррекции не требует еще одной коррекции.

Таким образом, при сложении двоично-десятичных чисел выполняется коррекция результата по правилу: если был межтетрадный перенос (переполнение тетрады) или получилась запрещенная комбинация, то к этой тетраде добавляется + 6 (0110).
Еще один пример: A = 479; B = 128; $C = A + B$ = 607.

0100	0111	1001	
+ 0001	0010	1000	
0101	1010	0001	– выполняем коррекцию;
0000	0110	0110	
0110	0000	0111	– результат верный.

В старшей тетраде коррекция 0, в средней + 6 (запрещенная комбинация), в младшей тетраде + 6 (был перенос).

ПЕРЕПОЛНЕНИЕ РАЗРЯДНОЙ СЕТКИ МАШИНЫ

При сложении чисел одинакового знака может возникнуть переполнение разрядной сетки. Признаком переполнения служит отличие знака результата от знаков слагаемых. Пусть, например, требуется сложить два числа:

$$A = -1101 \qquad [A]_{\text{доп}} = 10011;$$
$$B = -1001 \qquad [B]_{\text{доп}} = 10111;$$
$$C = A + B \qquad [C]_{\text{доп}} = 01010.$$

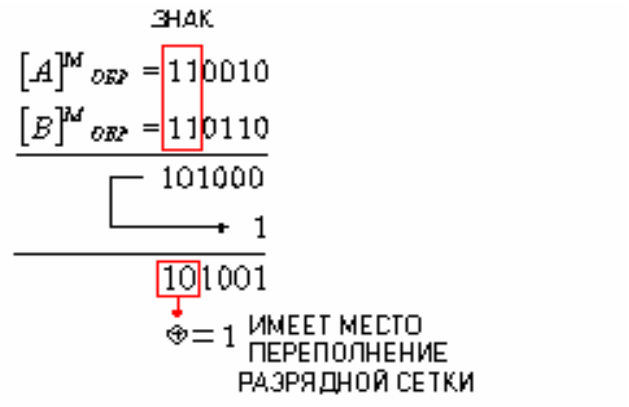
Складывали два отрицательных числа, а ответ – положительный. Чтобы не было переполнения, необходимо расширить разрядную сетку. Запишем числа A и B в 5 битах:

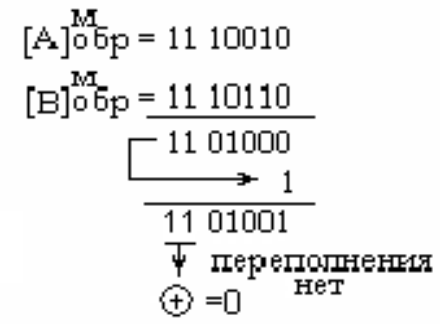
$$A = -01101 \qquad [A]_{\text{доп}} = 110011;$$
$$B = -01001 \qquad [B]_{\text{доп}} = 110111;$$
$$C = A + B \qquad [C]_{\text{доп}} = 101010;$$
$$[C]_{\text{обр}} = 101001;$$
$$C = -10110.$$

Теперь ответ верный. В вычислителях для контроля переполнения следят за переносом в знаковый разряд и из него. Если оба переноса имеют место или оба отсутствуют, то переполнения нет. Если есть только один из переносов, то имеет место переполнение разрядной сетки.

В некоторых машинах для контроля переполнения используют так называемые модифицированные коды: прямой, обратный, дополнительный. В этих кодах под знак числа отводится по 2 бита. После сложения эти знаковые биты складывают между собой по модулю два. Если результат сложения равен 0, то нет переполнения, если результат сложения равен 1, то переполнение есть. Проверим это правило на примере обратного модифицированного кода. Сложим два числа:

$$A = -1101;$$
$$B = -1001;$$





Таким образом, модифицированные коды удобны для использования, хотя и занимают на 1 бит больше памяти.

КОНТРОЛЬ ИНФОРМАЦИИ (МОЖНО НЕ ЧИТАТЬ СМ. КАК УСПЕВАЮТ)

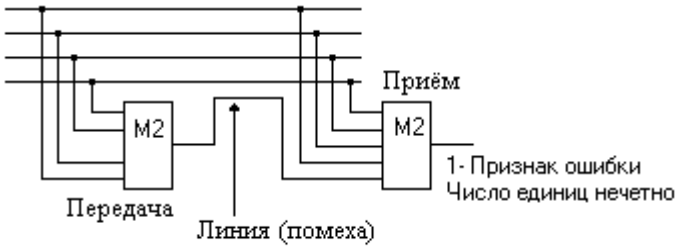
Рассмотренные ранее алгоритмы выполнения операций дают правильный результат, если не будет сбоев в работе машины. Если произойдет сбой, то ответ будет неправильный и пользователь никогда об этом не узнает, поэтому в первых вычислительных машинах всегда выполнялся двойной счет. В современных машинах существуют специальные системы контроля, которые позволяют определить наличие ошибки и даже автоматически ее исправить. Вообще, система контроля это совокупность средств и методов, обеспечивающих проверку правильности работы устройства, обнаружение и исправление ошибок.

Простейший метод контроля – проверка на четность (проверка паритета). Исходное число дополняют битом четности, в который заносят 0 или 1 с тем, чтобы общее число единиц было четным (рис. 2.8).

Число	Бит паритета	Код
1010	0	10100
1000	1	10001
0010	1	00101

Добавление бита паритета

Этот метод позволяет выявить только одну ошибку. Кодировка и проверка выполняются автоматически с помощью сумматоров по модулю два (рис).



Проверка паритета в месте приема

Для автоматического исправления ошибки этот метод видоизменяют – добавляют несколько контрольных разрядов по типу матрицы (рис. 2.10).
Здесь M – информационные разряды;

K – контрольные разряды;

$N = M + K = 9 + 6 = 15$ – общее число разрядов кода.

Проверку на четность выполняют по строкам и по столбцам. Одиночная ошибка сразу выявляется и исправляется

M_1	M_2	M_3	K_1
M_4	M_5	M_6	K_2
M_7	M_8	M_9	K_3
K_4	K_5	K_6	

Добавление битов паритета в матрицу

0	1	1	0
1	0	1	0
0	0 $\rightarrow 1$	0	1?
1	0 ?	0	

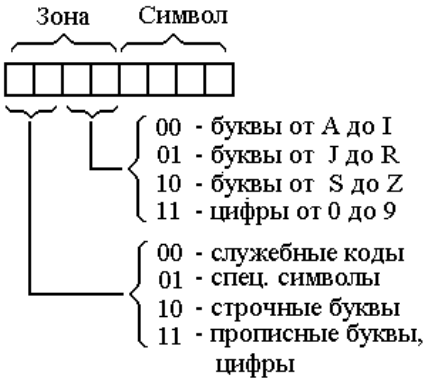
. Исправление ошибки в матрице

Кодирование по методу четности – довольно мощный и надежный способ защиты информации. Такие узлы находят применение на практике.

Для более длинных посылок или обнаружения двух и более ошибок используют и более сложные коды, например, коды Хемминга.

ПРЕДСТАВЛЕНИЕ АЛФАВИТНО-ЦИФРОВОЙ ИНФОРМАЦИИ

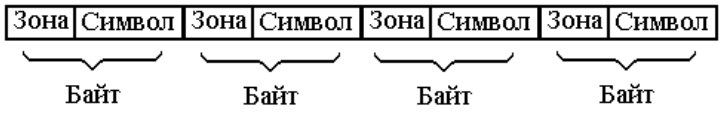
С помощью двоичных знаков можно записывать, хранить и обрабатывать не только числа, но и любую символьную информацию. Для этого каждому символу необходимо присвоить свой код. С помощью кода, состоящего из *n*-разрядов, можно представить 2^n различных символов. В 1963 году был введен 7-разрядный код ASCII (American Standard Code for Information Interchange – американский стандартный код для обмена информацией). Он позволял кодировать 128 символов. На основе этого кода построен отечественный код КОИ-7 и восьмиразрядные коды КОИ-8 и ДКОИ (двоичный код обработки информации), в которых каждый символ кодируется 1 байтом. Последующие модификации кода ASCII также 8-разрядные (байтовые). Поэтому можно закодировать 256 различных символов. Например, разряды байта кода ДКОИ распределяются так, как показано на рис.



Признаки символов в байте кода ДКОИ

Старшая тетрада (зона) носит служебный характер и указывает тип символа. Код непосредственно символа заносится в младшую тетраду. В других кодах (КОИ-8 и ASCII) кодировка зоны отличается от приведенной на рис. 2.12, но смысл зоны сохраняется – признак символа.

Любая информация устройством ввода преобразуется в код и хранится в памяти машины в виде слов (рис. 2.13).



Формат слова (4 байта)

Размер слова может быть различным: 2 байта, 4 байта, 8 байт и т.д.

Любая деловая информация содержит цифр больше, чем букв, поэтому для записи чисел используются специальные форматы: *двоичные* и

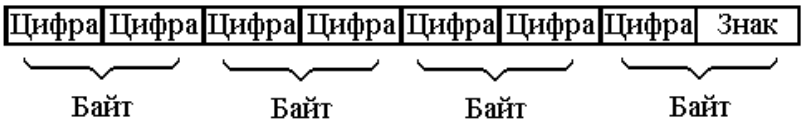
десятичные форматы целых чисел (с фиксированной запятой) и *форматы с плавающей запятой*. Кратко ознакомимся с ними.

Десятичный формат. Используется для записи целых чисел, при этом знак числа записывают в зону младшего байта. Это так называемый десятичный распакованный формат – формат с зоной (рис. 2.14).

Очевидно, что память используется не рационально, так как хранить одинаковую зону для каждой цифры неразумно, поэтому используют десятичный упакованный формат (рис. 2.15). Здесь код знака

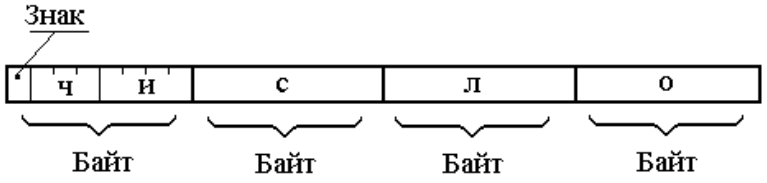
Десятичный распакованный формат

числа содержится в младшей тетраде младшего байта. Так записываются положительные и отрицательные числа. Память используется более рационально, но это ***только десятичные числа***.



Десятичный упакованный формат

Двоичный формат. Используется для записи чисел с фиксированной и с плавающей запятой. Формат с фиксированной запятой для целых чисел приведен на рис. 2.16.



Двоичный формат для целых чисел

Знак кодируется как обычно: 0 – плюс, 1 – минус. Число записывается в двоичной системе счисления, и каждый байт представляют парой шестнадцатеричных цифр. Максимальное положительное число, представимое в этом формате:

0111 1111 ….1111 или

$7\ F\ FF\ FF\ FF_{(16)}$.

Отрицательные числа в этом формате представляют в дополнительном коде. Так, числа +50 и –50 (десятичные) будут представлены в следующем виде:

00 00 00 32 (+50)

FF FF FF CE (–50).

Числа +10 и –10 (десятичные) записываются так:

00 00 00 0*A* (+10)

*FF FF FF F*6 (–10).

Какое наибольшее (по модулю) отрицательное число можно записать в этом формате? Для положительных чисел это $2^n - 1$, а для отрицательных 2^n. Возьмем, например, $n = 3$. Тогда 0 111 (или $2^3 - 1 = +7$). Дополнительный же код для числа – 7 = – 111 равен 1 001, т. е. дополнение не равно нулю и можно еще уменьшить отрицательное число с тем, чтобы дополнение стало равно нулю. Значит, в 3 битах можно записать число –8, а его дополнительный код равен 1 000.

Поэтому в формате 4 байта представимые целые числа, лежат в диапазоне от -2^{31} до $+2^{31} - 1$, т. е. диапазон несимметричный:

$$-2147483648 \le N \le +2147483647$$

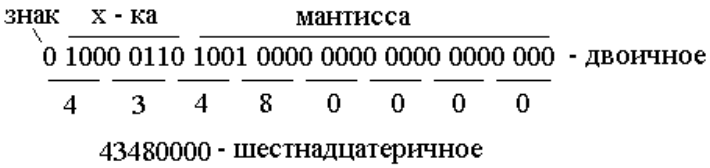
$$-80\ 00\ 00\ 00 \le N \le +7F\ FF\ FF\ FF.$$

В двоичном формате с плавающей запятой положительные числа записываются в нормализованном виде (см. раздел 2.4) в прямом коде, а отрицательные числа (мантиссы) записываются в дополнительном коде. Характеристики чисел всегда положительны. Например, десятичные числа +50 и –50 в формате с плавающей запятой будут записаны так:

$A = \pm 50 = \pm 32_{16} = \pm 0011\ 0010_2 = \pm 0,110010\ 2^{+0110}$.

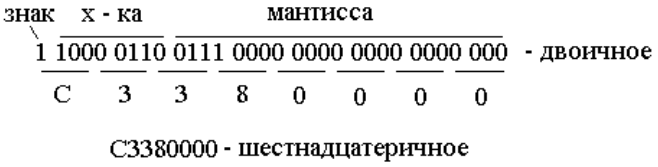
Характеристика числа $X = 128 + q = 128 + 6 = 134_{10} = 86_{16} = 1000\ 0110_2$.

Число +50 записывается так, как показано на рис.



Число +50 в формате с плавающей запятой и скрытой единицей мантиссы

Число –50 записывается так, как показано на рис.



Число –50 в формате с плавающей запятой и скрытой единицей мантиссы