Министерство образования и науки Украины
Харьковский национальный университет радиоэлектроники

Шкиль А.С., Сыревич Е.Е., Кораблёв Н.М.

АПВТ

1. Способы представления чисел в ЭВМ

СКС

Харьков
2009

Содержание

Теория

1.1 Системы счисления

Компьютерная Арифметика (**КАР)** – система принципов и форм представления числовой информации, методов и алгоритмов выолнения арифметических операций и вычисления элементарных функций, рассматриваемых на уровне структурной внутренней организации технических средств компьютерных систем.

Эта часть вычислительной математики, ориентирована на логический уровень вычислительных структур и процессов в них.

Алгоритм – способ преобразования информации, который задается с помощью конечной системы правил.

Информация представляется в виде совокупности цифр (чисел) в некоторой **системе счисления (СС)**; сами же цифры отображаются сигналами, имеющими конечное число уровней квантования.

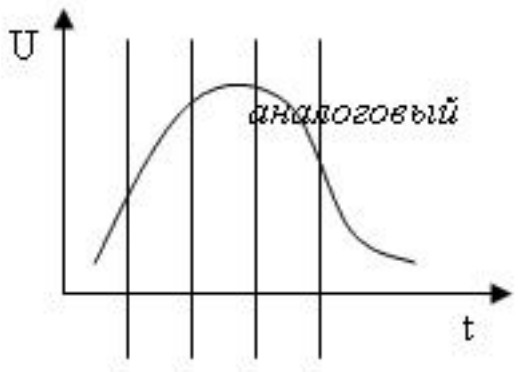


Рисунок 1.1. Представление информации.

СС - совокупность приёмов и правил для установления однозначного соответствия между числом и его представлением в виде некоторой совокупности знаков (символов).

Количественный эквивалент числа (КЭЧ) - некоторое количество, однозначно соответствующее числу. Это абстрактно-интуитивный уровень.

Каждой цифре в записи числа сопоставляется некоторое, выражаемое этой цифрой и называемое **количественным эквивалентом цифры (КЭЦ).**

Пример КЭЦ:

151 кг = 1ц + 50кг + 1кг

Системой счисления называется совокупность цифровых знаков и правил их записи, применяемая для однозначного представления чисел. Системы счисления подразделяются на позиционные и непозиционные.

Непозиционными называются такие системы, в которых применяется неограниченное количество цифр, причем значение каждой цифры не зависит от ее позиции в числе. Примером непозиционной является римская система счисления. Непозиционные системы в настоящее время используются редко, в основном для целей нумерации:

Римская СС: MMIV - 2004, MCMLXXXIX - 1989.

I - 1, V - 5, X - 10, L - 50, C - 100, D - 500, M - 1000.

Позиционными называются такие системы, в которых применяется ограниченный набор цифр, причем значение каждой цифры находится в строгой зависимости от ее позиции в числе. Количество различных цифр, применяемых в данной системе, называется ее *основанием*:

Десятичная, двоичная, восьмеричная, шестнадцатеричная системы счисления.

Недостатки непозиционных СС:

- теоретически неограниченный набор цифр;
- сложность технической реализации обрабатывающих устройств;
- отсутствие наглядности в представлении чисел;
- отсутствие однозначности.

Покажем некоторые свойства позиционных систем на примере **десятичной** системы. В ней применяется 10 цифр: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 , поэтому ее основанием является число 10. Произвольное десятичное число X_1 = 118,375 можно представить в следующем виде:

$$118,375 = 1\cdot 10^2 + 1\cdot 10^1 + 8\cdot 10^0 + 3\cdot 10^{-1} + 7\cdot 10^{-2} + 5\cdot 10^{-3}.$$

В целой части равенства записано символическое изображение числа. Правая часть равенства показывает, что все цифры числа в разных позициях имеют разный *вес*. Каждая позиция с присвоенным ей номером и весом называется *разрядом* числа. В частности, единица в старшем разряде означает сотню, а единица в следующем разряде – только десяток. Анализ структуры числа X_1 показывает, что любое десятичное число может быть представлено в виде суммы попарных произведений:

$$X = \sum_{i=-F}^{J-1} x_i \cdot 10^i, \qquad (1.1)$$

где X_i = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 } – цифра данной системы;

10^i – разрядный вес этой цифры;

J – количество разрядов целой части числа (до запятой);

F – количество разрядов дробной части числа (после запятой).

В определении позиционных систем счисления не наложено никаких ограничений на величину основания. Отсюда очевидно, что основанием системы счисления может быть не только число 10, но и любое другое целое число. При этом структурно некоторое число в другой системе счисления также будет состоять из суммы попарных произведений цифр и степеней основания системы. Если *основанию* любой системы счисления *присвоить*, по аналогии с числом десять, *обозначение 10*, то формула (1.1) будет справедливой для записи чисел в любой системе счисления.

В частности, можно, например, представить **двоичную** систему счисления, основанием которой является число 2, и следовательно для изображения числа применяется всего две цифры – ноль (0) и единица (1), т.е $X_i = \{0,1\}$.

Основание системы счисления – 2, можно записать с помощью имеющихся цифр только одним способом, т.е. в виде числа 10. Таким образом, любое число в двоичной системе записывается в виде комбинации нулей и единиц, расставленных согласно формуле (1.1). Так, рассмотренное выше десятичное число 118,375 в двоичной системе запишется следующим образом: $X_1 = 118,375_{10} = 1110110,011_2$, или:

$$1110110,011_2 = 1\cdot 2^6 + 1\cdot 2^5 + 1\cdot 2^4 + 0\cdot 2^3 + 1\cdot 2^2 + 1\cdot 2^1 + 0\cdot 2^0 + 0\cdot 2^{-1} + 1\cdot 2^{-2} + 1\cdot 2^{-3} =$$

$$= 118,375_{10}$$

Нижний индекс справа от числа, показывает величину основания системы счисления, в которой записано данное число. Используя формулу (1.1), можно записать это же число в **восьмеричной** системе счисления.

$$X_1 = 166,3_8 = 1\cdot 8^2 + 6\cdot 8^1 + 6\cdot 8^0 + 3\cdot 8^{-1} = 64 + 48 + 6 + 3/8 = 118,375_{10}$$

Анализ формулы (1.1) позволяет установить основные свойства позиционных систем счисления.

1. Минимальным, отличным от нуля числом, является $X_{\min} = 10^{-F}$, которое численно равно весу самого младшего разряда. Данное число носит название "единицы самого младшего разряда" (единицы СМР) или "минимального шага приращения" (для целых чисел при *F* = 0, $X_{\min} = 1$).

2. Максимальным в любой позиционной системе счисления является число $X_{\max} = 10^{J} - 10^{-F}$, когда во всех разрядах записана самая старшая цифра системы (для целых чисел при *F* = 0, $X = 10^{J} - 1$).

3. Общее количество чисел, которые можно записывать в *n-разрядную сетку* (*n* = *J* + *F*), равно $N = 10^{n}$ (следует из формулы 1.1).

Соответственно, если известно *N*, то количество разрядов, обеспечивающее запись любого числа из множества *N*, равно $n = \log_{10} N$.

Например, в один байт (8 двоичных разрядов) можно записать $2^{8} = 256$ двоичных чисел (основание системы счисления – 2) от 00000000 до 11111111.

В техническом аспекте длина числа интерпретируется часто как *длина разрядной сетки* (ДРС).

Для различных СС характерна различная ДРС, необходимая для записи одного и того же числа.

1.2 Перевод чисел из одной системы счисления в другую

Задача перевода чисел из одной СС в другую является одной из главных задач в КАР.

Ее можно сформулировать следующим образом:

1. Требуется перевести некоторое число X, записанное в позиционной однородной СС с основание k_1 в такую же СС, имеющее основание k_2.

2. По изображению операнда X в СС с нованием k_1 найти Y того же операнда в СС с основанием k_2.

Существует две группы методов перевода чисел:

- табличные;
- расчетные.

При переводе чисел между системами счисления может использоваться, как арифметика исходной системы счисления, так и арифметика новой системы счисления. Если используется арифметика новой системы счисления, то работа по переводу числа осуществляется в

соответсвии с формулой 1.1.

"Расчётные методы"

Перевод целых чисел

Перевод из десятичной системы:

Обычно для перевода числа из десятичной системы счисления в произволную позиционную систему счисления используется метод деления. Этот метод заключается в последовательном делении переводимого числа на основание новой системы счисления до тех пор, пока не получится частное, у которого целая часть равна 0. Число в новой системе счисления записывается из остатков от последовательного деления, причем последний остаток будет старшей цифрой нового числа. Вычисления, осуществляемые по этому правилу, должны выполняться в исходной системе счисления. Для перевода числа *Nисх* с основанием исходной системы счисления $B_{исх}$ в число $N_{нов} = z_{n-1}z_{n-2}\dots z_1 z_0$ с новым основанием системы счисления $B_{нов}$ в данном случа используется **метод деления**.

- В начале, $N^{(1)}_{исх} := N_{исх}$.
- На *i*-ом шаге вычисляется $N^{(i+1)}_{исх}$ и цифра z_{i-1}.
- На первом шаге: $N^{(1)}_{исх} / B_{нов} = N^{(2)}_{исх}$. Остаток z_0. Замечание: $z_0 < B_{нов}$.
- На *k*-ом шаге: $N^{(k)}_{исх} / B_{нов} = N^{(k+1)}_{исх}$. Остаток z_{k-1}.
- На *i*-ом месте находится z_{i-1} - остаток от деления $N^{(i)}_{исх} / B_{нов}$.

Пример: $1020_{10} = ?_8$

$B_{исх} = 10,$
$B_{нов} = 8,$
$N_{исх} = 1020 = z_3 z_2 z_1 z_0.$

Таблица 1.1

Шаг 1:	1020	:	8	=	127	Остаток: 4	127	·	8	=	1016
Шаг 2:	127	:	8	=	15	Остаток: 7	15	·	8	=	120
Шаг 3:	15	:	8	=	1	Остаток: 7	1	·	8	=	8
Шаг 4:	1	:	8	=	0	Остаток: 1					

$\rightarrow \mathbf{1020}_{10} = 1774_8.$

Перевод в десятичную систему:

Метод умножения используется, в основном, для перевода числа в десятичную систему счисления. Этот метод заключается в сложении результатов произведений разрядов исходного числа на основание исходной системы счисления возведенное в степень соответствующую позиции данного разряда, вычисленных в десятичной системе счисления. Для перевода числа $N_{исх}$ с основанием исходной системы счисления $B_{исх}$ в число $N_{нов} = z_{n-1}z_{n-2}\ldots z_1 z_0$ с новым основанием системы счисления $B_{нов}$ в данном случае используется **метод умножения**.

● Используется следующая схема для представления чисел:

$$N_{исх} = (\ldots(((z_{n-1}\cdot B_{исх} + z_{n-2})\cdot B_{исх} + z_{n-3})\cdot B_{исх}) + z_0)$$

Метод **умножения**:

● Необходимо *n* шагов для выполнения операции: $N^{(1)}{}_{нов} = z_{n-1}.$

● На *i*-ом шаге: $N^{(i+1)}{}_{нов} = N^{(i)}{}_{нов}\cdot B_{исх} + z_{n-(i+1)}$

$$N_{нов} = N^{(n)}{}_{исх}.$$

Пример: $3FC_{16} = ?_{10}$

Таблица 1.2

Шаг 1:	$N^{(1)}{}_{10} = 3$
Шаг 2:	$N^{(2)}{}_{10} = N^{(1)}{}_{10}\cdot 16 + 15 = 3\cdot 16 + 15 = 63$
Шаг 3:	$N^{(3)}{}_{10} = N^{(2)}{}_{10}\cdot 16 + 12 = 63\cdot 16 + 12 = 1020$

$\rightarrow \mathbf{3FC}_{\mathbf{16}} = \mathbf{1020}_{\mathbf{10}}.$

Перевод чисел между системами счисления с основанием, которое является степенью двойки:

Поскольку в настоящее время помимо двоичной системы счисления распространено использование только двух систем счисления с основанием, являющимся степенью двойки (8 и 16), то и примеры приводятся только для них:

$2^n \to 2$

--> Двоичное представление цифр в системах счисления

Пример: восьмеричная --> двоичная:

3 двоичных цифры вместо 1 восьмеричной цифры

шестнадцатеричная --> двоичная:

4 двоичных цифры вместо 1 шестнадцатеричной цифры

Пример: $1F2_{16} = 0001\ 1111\ 0010_2$

2. Перевод $2 \to 2^n$

--> "сжатие" цифр

Пример: $101\ 111\ 010_2 = 572_8$

3. Перевод $2^n \to 2^m$

$\to 2^n \to 2 \to 2^m$

Пример: восьмеричная --> шестнадцатеричная

$$\begin{aligned} 6351_8 &= 110\ 011\ 101\ 001_2 \\ &= 1100\ 1110\ 1001_2 \\ &= CE9_{16} \end{aligned}$$

Поскольку *точный* перевод дробных чисел $(R_{исх} \to R_{нов})$ не всегда возможен, то искать результат следует в виде: $R_{нов} = R_{исх} + e$, где *e* "достаточно" мало.

Перевод из десятичной системы.

При переводе из десятичной системы дробных чисел используется метод умножения (в

отличие от перевода целых чисел):

$$R^{(0)}{}_{нов} := R_{нов} = (z_1 + (z_2 + \ldots + (z_{m-1} + z_m \cdot B^{-1}{}_{нов}) \cdot B^{-1}{}_{нов} \ldots) \cdot B^{-1}{}_{нов}) \cdot B^{-1}{}_{нов}$$

Перевод осуществляется за ***k*** шагов, где ***k*** - требуемая точность.

На ***i***-ом шаге вычисляется $R^{(i)}{}_{нов} + z_i = R^{(i-1)}{}_{нов} \cdot B_{нов}$

Пример: $0.19_{10} = ?_2$, при $\boldsymbol{k} = 9$

$$R^{(i-1)}{}_{нов} \cdot B_{нов} = R^{i}{}_{нов} + z_i$$

Таблица 1.3

Шаг	$R^{(i-1)}{}_2$	Операция	$R^{(i)}{}_2$	z_i
1	0.19	0.19 · 2 = 0.38	0.38	0
2	0.38	0.38 · 2 = 0.76	0.76	0
3	0.76	0.76 · 2 = 1.52	0.52	1
4	0.52	0.52 · 2 = 1.04	0.04	1
5	0.04	0.04 · 2 = 0.08	0.08	0
6	0.08	0.08 · 2 = 0.16	0.16	0
7	0.16	0.16 · 2 = 0.32	0.32	0
8	0.32	0.32 · 2 = 0.64	0.64	0
9	0.64	0.64 · 2 = 1.28	0.28	1

$$\rightarrow \mathbf{0.19_{10} = 0.001100001_2 + e}$$

Перевод в десятичную систему.

При переводе в десятичную систему дробных чисел используется метод деления (в отличие от перевода целых чисел). Перевод осуществляется за m-1 шагов:

$$R_{исх} = (z_1 + (z_2 + \ldots + (z_{m-1} + z_m \cdot B^{-1}{}_{исх}) \cdot B^{-1}{}_{исх} \ldots) \cdot B^{-1}{}_{исх}) \cdot B^{-1}{}_{исх}$$

$$R^{(0)}{}_{нов} = z_m \cdot B^{-1}{}_{исх}$$

$$R^{(i)}_{нов} = (R^{(i-1)}_{нов} + z_{m-i}) \cdot B^{-1}_{исх}$$

Пример: $0.11001_2 = ?_{10}$

$$R^{(0)}_{10} = z_5 : 2 = 1 : 2 = 0.5$$
$$R^{(1)}_{10} = (R^{(0)}_{10} + z_4) : 2 = (0.5 + 0) : 2 = 0.25$$
$$R^{(2)}_{10} = (R^{(1)}_{10} + z_3) : 2 = (0.25 + 0) : 2 = 0.125$$
$$R^{(3)}_{10} = (R^{(2)}_{10} + z_2) : 2 = (0.125 + 0) : 2 = 0.5625$$
$$R^{(4)}_{10} = (R^{(3)}_{10} + z_1) : 2 = (0.5625 + 0) : 2 = 0.78125$$

$$\rightarrow \mathbf{0.1101_2 = 0.78125_{10}}$$

"Табличный метод"

Табличный метод в простейшем случае предполагает хранение в памяти ЭВМ таблицы соответствий между всеми числами в СС с основаниями k_1 и k_2, а сама процедура перевода сводится к обращению к этой таблице.

Пример:

Таблица 1.4

Десятичное число	k=2	k=3	k=8
0	0000	000	00
1	0001	001	01
2	0010	002	02
3	0100	010	03
4	0101	011	04
5	0110	012	05
6	0111	020	06
7	1000	021	07
8	1000	022	10
9	1001	100	11

Достоинством метода является высокая скорость перевода. Но недостатком является то, что размер таблицы и занимаемый ею объём памяти очень большой, что технически неприемлемо.

1.3 Представление чисел с фиксированной и плавающей точкой

Система машинных чисел

Система вещественных чисел используется в ручных расчетах, предполагается быть бесконечной и непрерывной, т.е. в ней нет ограничений на диапазон и точность чисел.

В компьютерной технике разрядная сетка устройств имеет ограничения – диапазон чисел и точность их представления.

Система машинных чисел является дискретной, образуя подмножество системы вещественных чисел.

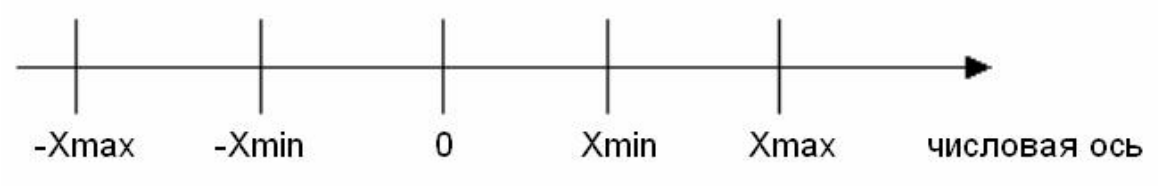


Рисунок 1.3. Система машинных чисел.

X_{max}, X_{min} – max и min представления числа, между которыми находится конечное множество представимых чисел.

Если результат больше x_{max}, то возникает переполнение (overflow).

Если модуль результата меньше x_{min}, то фиксируется антипереполнение (underflow).

Большинство компьютеров при антипереполнении возвращают ноль. Поэтому область от $-X_{min}$ до X_{min} за вычетом истинного нуля называется **областью машинного нуля**.

В зависимости от назначения и конструкции ЭВМ в них применяются две формы представления двоичных чисел: естественная и нормальная.

Естественной называется такая форма числа, которая в неявном, условном виде реализует формулу, приведенную ниже, т.е. число записывается только с помощью набора значащих цифр x_j без явного указания их весов и знаков сложения между ними. Отсчет разрядов (что равносильно указанию их весов) ведется от точки, которая обычно фиксируется между целой и дробной частями числа. Изображение числа имеет следующий вид:

$$X = \sum_{i=-F}^{J-1} x_j \cdot 10^i = x_{j-1}x_{j-2}\ldots x_2x_1x_0x_{-1}x_{-2}\ldots x_{-F}, \quad (1.2)$$

этой формой мы чаще всего пользуемся в повседневной жизни.

Нормальной называется такая форма числа, которая в неявном, условном виде реализует формулу:

$$X = \underbrace{10^p}_{\text{порядок}} \underbrace{\sum_{i=-F}^{J-1} x_j \cdot 10^i}_{\text{мантисса}} \quad (1.3)$$

когда число представляется как произведение некоторой целой степени основания системы и цифровой части, являющейся правильной дробью. При этом показатель степени основания называется *порядком*, а цифровая часть – *мантиссой* числа. Мантисса может иметь знак. Знак мантиссы – это знак всего числа. При записи и порядок, и мантисса представляются в естественной форме, как показано в таблице:

Таблица 1.5

Система счисления	Форма записи	
	естественная	нормальная
Десятичная	118.375	$10^3 \cdot 0.118375 = 10^4 \cdot 0.0118375 = 10^{-3} \cdot 118375 =$ $= 10^2 \cdot 1.18375 = \ldots$
Двоичная	1110110.011	$10^{111} \cdot 0.1110110011 = 10^{1000} \cdot 0.01110110011 = \ldots$

При записи числа в нормальной форме достаточно указать только порядок и мантиссу, не фиксируя в явном виде основание системы 10. Например, можно (в машинах так и делается) записать $10^3 \cdot 0.118375 = 0.118375e3$. Формула, записанная выражением 1.3, описывает *нормальную* (как уже было сказанно), *научную* или *полулогарифмическую* запись числа, т.к. половина числа записано нормально, а половина логарифмически.

Поскольу в естественной форме положение точки в числе строго зафиксировано между целой и дробной частями, то числа в этой форме называют *числами с фиксированной точкой*. Соответственно и машины, оперирующие числами в естественной форме, называются

машинами с фиксированной точкой.

В определении нормальной формы не наложено никаких ограничений на величину мантиссы, как праило мантисса должна быть правильной дробью. Поэтому положение точки в мантиссе может изменяться при соответствующем изменении величины порядка. Точка при этом как бы плавает. Поэтому часто числа в нормальной форме называют *числами с плавающей точкой*, а вычислительные машины, использующие эту форму чисел, – *машинами с плавающей точкой*.

Под **разрядной сеткой** понимается определенное количество разрядов, выделенных для представления числа, а также разбиение их на упорядоченные группы для представления отдельных частей числа (таких как знак мантиссы, порядок и т.д.). Ниже, при рассмотрении вопроса о представлении чисел изображен пример двух разрядных сеток.

При разработке новой машины инженер-проектировщик должен четко знать, в каких случаях нужно закладывать в проект естественную, а в каких случаях - нормальную форму представления чисел. Поэтому необходимо провести анализ характеристик обеих форм. Рассмотрение приведенной выше таблицы позволяет отметить следующие два недостатка машин, работающих с числами в нормальной форме.

Первый недостаток заключается в необходимости усложнения арифметического устройства из-за возможной неоднозначности записи числа. Из всего множества изображений числа в нормальной форме то изображение, в котором старший разряд мантиссы (первая цифра после точки) не равен нулю, называется *нормализованным числом*. Соответственно все остальные изображения этого же числа называются *ненормализованными*.

В памяти машины все числа хранятся только в нормализованном виде, так как в противном случае из-за ограниченности разрядной сетки терялись бы младшие разряды мантиссы, т.е. уменьшалась бы точность представления чисел. Но при арифметических действиях результаты могут получиться и ненормализованными (разрядная сетка регистра для результатов вычислений делается несколько длиннее, чем сетка у ячеек памяти). Поэтому в машинах предусмотрена специальная схема, которая автоматически производит нормализацию тех результатов, которые получились в процессе вычислений ненормализованными. Нормализация заключается в сдвиге мантиссы влево на столько разрядов, сколько у нее было нулей после запятой, и в одновременном уменьшении порядка числа на такое же количество единиц.

Пример:

В результате нормализации числа $X_1 = 10^5 \cdot 0.00118375_{10}$ получаем $X_{1\,\text{норм}} = 10^3 \cdot 0.118375 = 0.118375\text{e}3_{10}$.

В результате нормализации числа $X_2 = 10^{1000} \cdot 0.0111011011_2$ получаем $X_{2\,\text{норм}} = 10^{111} \cdot 0.111011011 = 0.111011011\text{e}111_2$.

Второй недостаток заключается в необходимости усложнения структуры АЛУ машины вследствие того, что в нем приходится производить разные действия с разными частями чисел

(мантиссами и порядками). Так при умножении двух чисел требуется мантиссы операндов перемножить, а их порядки алгебраически сложить.

Машина, работающая с числами в естественной форме, не имеет указанных недостатков. Однако это не значит, что машины с фиксированной точкой обладают только положительными характеристиками. Эти машины имеют также довольно крупные недостатки, связанные в основном с малым диапазоном чисел, с которыми они могут оперировать при ограниченной разрядной сетке.

Рассмотрим этот вопрос более подробно. Допустим, что имеется машина с девятиразрядной сеткой для записи десятичных (для простоты рассмотрения) чисел. Пусть также задано, что машина может работать в двух режимах: с фиксированной и плавающей точкой, причем на дробную часть числа в первом случае и на мантиссу во втором отводится пять разрядов. Тогда разрядная сетка распределится между частями числа следующим образом:

Таблица 1.6

I. В режиме с фиксированной точкой	II. В режиме с плавающей точкой
Знак числа – 1 разряд,	Знак числа (мантиссы) – 1 разряд,
Целая часть – 3 разряда,	Мантисса – 5 разрядов,
Дробная часть – 5 разрядов,	Знак порядка – 1 разряд,
Итого – 9 разрядов.	Порядок числа – 2 разряда,
	Итого – 9 разрядов.
	

Слева изображена сетка условной машины в режиме с фиксированной точкой, справа – в режиме с плавающей точкой. Заметим, что знак числа имеет двоичную природу, так как может принимать только два значения: плюс или минус. В машинах принято плюс изображать нулем, а минус – единицей, хотя в принципе можно сделать и наоборот. Посмотрим, какие числа можно записать в любую из ячеек нашей машины:

Таблица 1.7

I. В режиме с фиксированной точкой	II. В режиме с плавающей точкой
$+\lvert X_{max}\rvert = +999.99999$	$+\lvert X_{max}\rvert = +0.99999 \cdot 10^{+99}$
.....	.....
.....	.....
$+\lvert X_{max}\rvert = +000.00001$	$+\lvert X_{max}\rvert = +0.10000 \cdot 10^{-99}$
000.00000 $-\lvert X_{max}\rvert = -000.00001$	0.00000 $-\lvert X_{max}\rvert = +0.10000 \cdot 10^{-99}$
.....	.....
.....	.....
$-\lvert X_{max}\rvert = -999.99999$	$-\lvert X_{max}\rvert = -0.99999 \cdot 10^{+99}$

Анализ правой части построенной таблицы показывает, что в режиме с плавающей точкой получается очень большой диапазон чисел $X_{маш.п.т.}$: от $|\pm 10^{+99}|$ до $|\pm 10^{-99}|$. Конечно, теоретически в результате вычислений может образоваться число, и не входящее в машинный диапазон. Однако практически такое событие маловероятно, а для некоторых инженерных, экономических и других задач вообще невозможно.

В режиме же с фиксированной точкой опасность того, что текущие машинные числа окажутся вне диапазона, вполне реальна, так как все машинные числа заключены в очень узких пределах: $-10^3 < X_{маш.ф.т.} < +10^3$. Поэтому в машинах с фиксированной точкой возможны два нежелательных случая: $|X_{маш.ф.т.}| < |X_{min}|$; $|X_{маш.ф.т.}| < |X_{max}|$. В первом случае любое число воспринимается машиной как нуль. Во втором случае получаются числа, превышающие по абсолютной величине максимальное машинное число, т. е. происходит переполнение разрядной сетки. При этом теряются старшие разряды числа, а это значит, что происходит его грубое искажение.

1.4 Кодирование чисел в ЭВМ

"Числа в прямом коде"

Вопрос о кодировании чисел возникает по той причине, что в машину нельзя либо нерационально вводить числа в том виде, в котором они изображаются человеком на бумаге. Во-первых, нужно кодировать знак числа. во-вторых, по различным причинам, которые будут рассмотрены ниже, приходится иногда кодировать и остальную часть числа.

Представление знаковых чисел

Для того, чтобы компьютер оперировал как положительными, так и отрицательными числами, в РС машинного представления числа необходимо ввести знаковую часть.

В двоичной СС знак представляется одним битом, принимающим значение «0» в случае положительного числа, и «1» - в случае отрицательного.

Обычно это крайний левый бит в представлении числа. Остальные - информационные разряды - представляют код модуля числа.

Т.о. компьютер имеет возможность манипулировать как знаковыми, так и беззнаковыми числами.

Для хранения числовых данных в КС используются устройства, называемые **регистрами.**

С учётом рассматриваемых ранее структур единиц информации вводят понятие формата данных.

Фактически регистр является аппаратной базой для представления числовых данных.

В компьютерах IBM PC AT машинное слово является 16-разрядным. Числовые данные могут иметь длину слова (16 бит) (word), полуслова (8 бит=байт) (byte), 32 бита - 2 слова (double word) - короткое целое (short integer).

Существует также 64 бита - учетверённое слово (guard word) - длинное целое (long int.).

В связи с тем, что алгоритмы выполнения компьютерных операций в КС имеют свою специфику, возникает проблема представления отрицательных чисел. Её решают за счёт особых способов кодирования модуля числа.

Замечание! Положение точки никогда явно не указывается, а только подразумевается.

Простейшим машинным кодом является прямой код, который получается при кодировании в числе только знаковой информации.

Прямым кодом отрицательного числа называется его изображение в естественной форме, у которого в знаковом разряде проставляется единица. Прямой код положительного числа совпадает с его обычным изображением в естественной форме.

Данное определение позволяет дать прямому коду такую интерпретацию:

$$[X]_{\text{пр}} = \begin{cases} X, & \text{при } X \geq 0 \\ 10^{J} + |X|, & \text{при } X < 0 \end{cases}$$

Приведенное выше аналитическое выражение показывает, что прямой код дробного числа (J = 0) формируется как сумма абсолютной величины исходного числа и единицы. Прямой код целого числа ($J = n$ - 1, причем знаковым является крайний левый разряд) формируется как сумма 10^{n-1} + | X |.

Пример. В прямом коде дробные двоичные числа $X_1 = +0.110011_2$ и $X_2 = -0.110011_2$ будут иметь вид $[X_1]_{пр} = 0.110011$; $[X_2]_{пр} = 1.110011$. Используя 8-разрядную сетку ($J = 7$), иными словами имея 8 разрядов для представления числа, в прямом коде целые числа $X_1 = +110011_2$ и $X_2 = -110011_2$ будут иметь вид $[X_1]_{пр} = 00110011$; $[X_2]_{пр} = 10110011$.

Прямой код получил широкое распространение в ЭВМ вследствие своей простоты. В нем удобно хранить числа в памяти, перемножать числа. Но он также обладает некоторыми недостатками. Во-первых при использовании прямого кода появляется возможность получения +0 и -0. Во-вторых, как оказалось, он плохо приспособлен для сложения чисел. Действительно, при алгебраическом сложении чисел в прямом коде требуется выполнить четыре действия:

1. Сравнить знаки слагаемых.

2. Сравнить слагаемые по модулю при неравенстве их знаков.

3. Выполнить соответствующую арифметическую операцию: сложение при равенстве знаков и вычитание из большего по модулю слагаемого меньшего при неравенстве их знаков.

4. Присвоить алгебраической сумме знак большего по модулю слагаемого.

Так как операция сложения значительно проще вычитания, то возникает вопрос: нельзя ли каким-то образом алгебраическое сложение свести к арифметическому? Оказывается, это возможно за счет несколько более сложного кодирования.

"Числа в обратном коде"

При замене операции вычитания сложением, помимо дополнения, можно использовать и другое вспомогательное число - инверсию отрицательного операнда, возникающую в результате замены его цифр взаимно обратными. Поскольку инверсия меньше дополнения на единицу младшего разряда, то для получения правильной разности необходимо производить соответствующую коррекцию суммы вспомогательных чисел.

Таким образом, *обратным кодом отрицательного двоичного числа* будем называть его дополнение по модулю до X_{max}, получаемое по следующему правилу: *в знаковом разряде проставляется единица, а во всех остальных разрядах цифры заменяются на взаимно обратные.*

Обратный код положительного числа совпадает с его прямым кодом. Аналитически обратный код определяется следующим соотношением:

$$[X]_{обр} = \begin{cases} X, & при\ X \geq 0 \\ X_{max} - X, & при\ X < 0 \end{cases}$$

С учетом цифры $X_{max} = 10^{J+1} - 10^{-F}$. Основное достоинство обратного кода по сравнению с дополнительным состоит в простоте процесса его формирования.

Пример. В обратном коде дробные двоичные числа $X_1 = +0.110011_2$ и $X_2 = -0.110011_2$ будут иметь вид $[X_1]_{обр} = 0.110011$; $[X_2]_{обр} = 1.001100$. Используя 8-разрядную сетку ($J = 7$) в дополнительном коде целые числа $X_1 = +110011_2$ и $X_2 = -110011_2$ будут иметь вид $[X_1]_{обр} = 00110011$; $[X_2]_{обр} = 11001100$.

"Числа в дополнительном коде"

Идея замены вычитания сложением основана на применении некоторых вспомогательных чисел, однозначно связанных с исходными отрицательными числами. Эти вспомогательные числа находятся как дополнение модулей заданных отрицательных операндов до некоторого *граничного числа* $X_{гр} = |X_{max}| + 10^{-F} = 10^{J}$:

$$[X_{доп}] = X_{гр} - [X]$$

Проиллюстрируем сказанное на простейшем примере с использованием десятичной системы счисления. Пусть имеется некоторая вычислительная машина, которая оперирует двухразрядными целыми числами, меньшими сотни: $0 \le X_{маш} \le X_{max} = 99$. Граничное число определяется как $X_{гр} = 99 + 1 = 100 = 10^2$. Далее требуется алгебраически сложить числа $X_1 = 84$ и $X_2 = -63$. В соответствии с вышеизложенным, второе слагаемое поскольку оно отрицательно, заменится дополнением до $X_{гр} = 10^2$, т. е. положительным числом $[X_2]_{доп} = 37$. Далее выполняется арифметическое сложение чисел X_1 и $[X_2]_{доп}$:

$$X_1 + [X_2] = 84 + 37 = 121$$

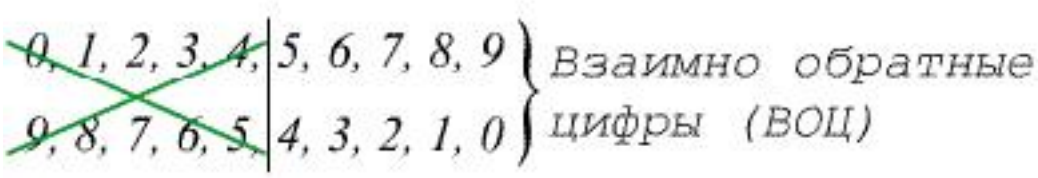
Поскольку числа не могут быть более сотни, то цифра в разряде сотен в сумме автоматически пропадает. Оставшуюся часть суммы можно считать результатом алгебраического сложения исходных чисел X_1 и X_2. Действительно, прямое вычитание подтверждает правильность конечного результата: $84 - 63 = 21$. Другого исхода не могло и быть, так как вначале в соответствии с приведенной выше формулой мы добавили ко второму слагаемому сотню, чтобы получить нужное дополнение

$$[X_2] = X_2 + X_{гр} = X_2 + 100 = -63 + 100 = 37,$$

а затем лишнюю сотню вычли из результата простым отбрасыванием появившейся и недопустимой в данной машине единицы в разряде сотен.

На первый взгляд кажется, что рассмотренный код не дает желаемого эффекта, так как нахождение дополнения отрицательного числа по приведенной выше формуле все равно происходит при помощи вычитания. Но это не так. Тот факт, что вычитание производится всегда из одного и того же и, главное, "круглого" числа, дает возможность избежать трудностей обычного вычитания.

Для дальнейшего изложения метода необходимо ввести некоторые новые понятия. Так *взаимно обратными* будем называть цифры, которые являются друг для друга дополнением до числа, на единицу меньшего, чем основание системы счисления. Чтобы проиллюстрировать сказанное, запишем в ряд все цифры используемой системы счисления, а затем под этим рядом запишем эти же цифры, но в обратном порядке, как это показано ниже:



Заметим одно замечательное свойство каждой пары цифр, расположенных относительно друг друга по вертикали при такой записи. Оказывается, что сумма любой из этих пар равна старшей цифре системы счисления. Поскольку такие пары образуются при записи всех цифр сначала в прямом, а затем в обратном порядке, будем называть цифры, дающие в сумме старшую цифру, *взаимно обратными*. Замена в некотором отрицательном числе всех цифр на взаимно обратные равносильна сложению исходного числа с $|X_{max}|$. Используя понятие взаимно обратных цифр, можно дать правило нахождения дополнения отрицательных чисел без использования вычитания: *для получения дополнения отрицательного числа следует все цифры исходного числа заменить на взаимно обратные и к полученному таким образом "обратному" числу (инверсному изображению) добавить единицу младшего разряда*.

Справедливость правила можно проиллюстрировать на рассмотренном выше примере. Действительно,

$$[X_2]_{доп} = X_2 + X_{гр} = X_2 + |X_{max}| + 1 = -63 + 99 + 1 = 36 + 1 = 37,$$

что и требовалось получить. Таким образом, в десятичной машине для образования дополнения любого числа достаточно иметь в памяти пять пар ВОЦ и уметь производить соответствующую замену в отрицательных числах.

Аналогично, вместе с тем значительно проще, эта процедура выполняется в машинах, использующих двоичную систему счисления. В этом случае нужна только одна пара ВОЦ: 0 и 1. Очевидно, что граничным числом в случае дробных чисел будет единица. Учитывая цифру знакового разряда, которая для отрицательных чисел всегда равна 1, следует считать $X_{гр}$ равным двум. В случае целых чисел граничное число всегда равно целой степени двух, равной

весу не существующего в данном числе разряда, расположенного слева от знаковой цифры.

Поскольку вспомогательные числа (дополнения) связанны с исходными отрицательными числами однозначным способом, то можно считать их заданными исходными числами, но некоторым образом закодированными. Таким образом, можно говорить о дополнительном коде машинных чисел и дать следующее его определение: *дополнительным кодом отрицательного двоичного числа* называется его дополнение до граничного числа. Оно получается по следующему правилу: *в знаковом разряде записывается единица, а во всех других разрядах цифры заменяются на взаимно обратные, после чего ко младшему разряду числа добавляется единица*.

Дополнительный код положительного числа совпадает с его прямым кодом.

Аналитически дополнительный код определяется следующим выражением:

$$[X]_{доп} = \begin{cases} X, & \text{при } X \geq 0 \\ X_{гр} - |X|, & \text{при } X < 0 \end{cases}$$

Пример. В дополнительном коде дробные двоичные числа $X_1 = +0.110011_2$ и $X_2 = -0.110011_2$ будут иметь вид $[X_1]_{доп} = 0.110011$; $[X_2]_{доп} = 1.001100 + 0.000001 = 1.001101$. Используя 8-разрядную сетку ($J = 7$) в дополнительном коде целые числа $X_1 = +110011_2$ и $X_2 = -110011_2$ будут иметь вид $[X_1]_{доп} = 00110011$; $[X_2]_{доп} = 11001101$.

Переход от дополнительного кода отрицательного числа к его прямому коду никаких трудностей не вызывает: нужно из исходного кода вычесть единицу младшего разряда, после чего во всех разрядах, за исключением знакового, заменить цифры на взаимно обратные. Однако практически производить такой переход легче по другому правилу: вначале заменить все цифры, за исключением знаковой, на взаимно обратные, после чего добавить единицу к младшему разряду.

Ниже представлена диаграмма, демонстрирующая на примере числа, записанные в дополнительном и прямом коде. Первые шестнадцать чисел начиная с нижней части диаграммы против часовой стрелки (0 : 15) кодируют идентичные числа в прямом и обратном коде. Если продолжать двигаться в том же направлении до нижней части диаграммы, то изображенные числа будут кодировать возрастающие числа в прямом коде (16 : 31). Если же двигаться от нижней части диаграммы по часовой стрелке до верхней части диаграммы, то проходимые таким образом двоичные числа будут кодировать в дополнительном коде уменьшающиеся числа (-1 : -16).

"5-разрядные числа в дополнительном коде"

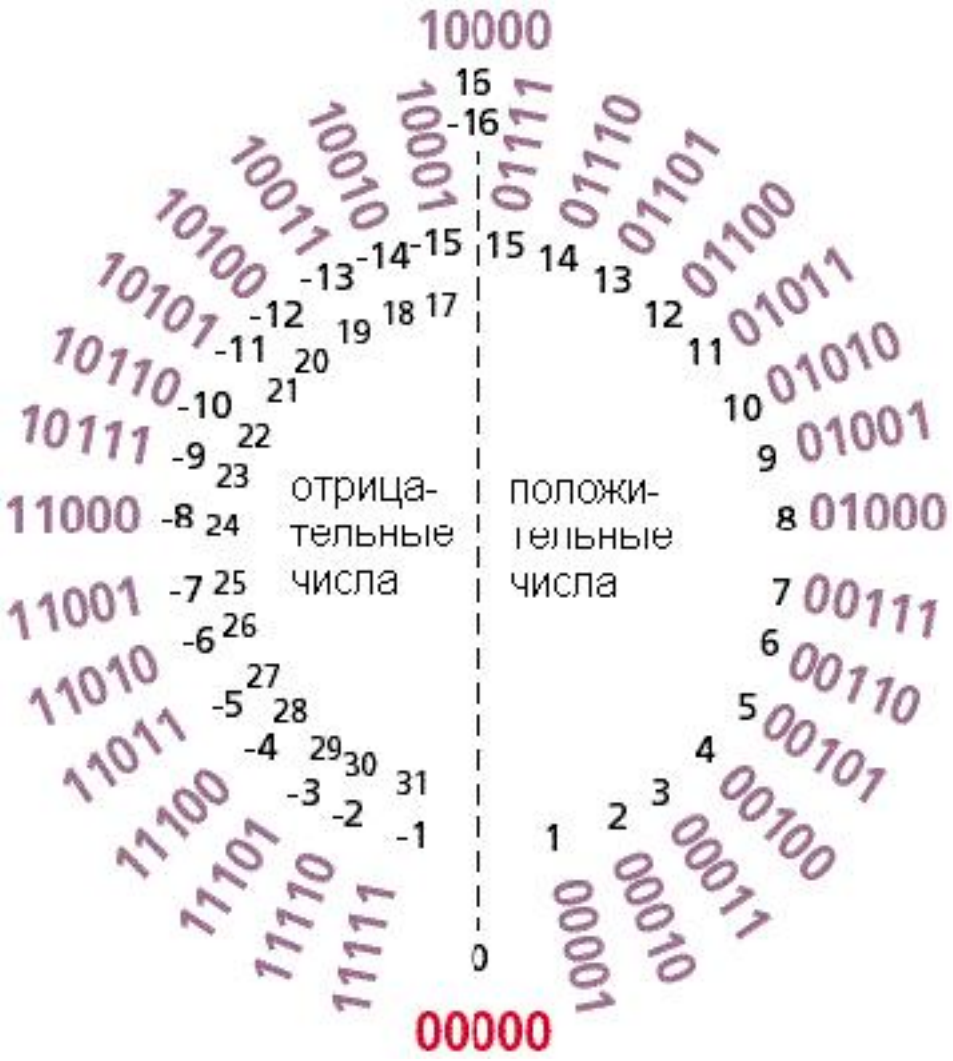


Рисунок 1.4. Модифицированные коды.

Модифицированные коды (МК) - коды, в которых знак изображается двумя одинаковыми цифрами. МК используются только в процессе выполнения операции. Знаковые разряды обрабатываются так же, как и обычные числовые разряды. Появление в знаковых разрядах МК разных цифр (01 - при сложении положительных чисел, 10 - при сложении отрицательных чисел) свидетельствует о ПРС.

МК позволяет формировать правильный знак результата даже в случае ПРС.

Пример:

$A = 00.111_2 = +7_{10}$

$B = 00.111_2 = +7_{10}$

$C = A + B$

(Правила сложения смотрите в разделе 3.1)

$$C = \begin{array}{r} {\scriptstyle 1\ 1\ 1} \\ 00{,}111 \\ +\ 00{,}111 \\ \hline 01{,}110 \end{array}$$

01 - это показатель некорректного выполнения операции, переполнения разрядной сетки.

Правильный вариант:

$$\begin{array}{r} {\scriptstyle 1\ 1\ 1} \\ 00{,}\mathbf{0}111 \\ 00{,}\mathbf{0}111 \\ \hline 00{,}1110 \end{array}$$

где 00 = +

1110 = 14 ⇒ полчен верный результат.

1.5 Критерии выбора СС

1. Простота технической реализации.

В техническом аспекте k означает количество устойчивых состояний вычислительного элемента. Чем меньше k, тем проще техническая реализация.

Пример: "вкл" – "выкл"

- транзисторы в ключевом режиме;
- рэле;
- диодные схемы;
- схемы магнитной памяти.

2. Наибольшая помехоустойчивость кодирования цифр.

Чем меньше k, тем система более помехоустойчива.

3. Минимальные затраты оборудования.

Введём параметр C:

C = k·n,

k - основание СС,

n - количество разрядов в числе.

$$C = k \cdot \log_k N$$

Необходимо найти оптимальное C (C_{opt}):

Решение указанного уравнения:

$$k_{opt} = e \approx 2.72 \Rightarrow k_{min} = 3$$
$$C_{min} = k_{opt} = e \cdot \ln N$$

$$C_{i(отн)} = \frac{C_i}{C_{opt}} => C_{i(отн)} = \frac{k_i \log_{k_i} N}{e \cdot \ln N} = \frac{k_i}{e \cdot \ln k_i}$$

k_i - заданное основание системы

Таблица 1.8

k_i	2	3	4	5	6
$C_{i(отн)}$	1,062	1,004	1,062	1,143	1,232

4. Простота арифметических действий.

Чаще всего для этой оценки используют операции сложения и вычитания.

Для любого k выполнение операций сложения и вычитания можно представить в виде:

В каждом разряде результата, полученного от сложения 2^k цифр x_i, y_i, получаем цифру результата q_i и цифру переноса p_i (перенос в старший разряд i+1) или заём p_i при выполнении операции вычитания, т.е. заём из старшего разряда.

Правила формирования цифр q_i и p_i можно представить следующей таблицей:

Таблица 1.9

Операция	Условие	q_i	p_i
Сложение	$x_i + y_i + p_{i-1} < k$	$x_i + y_i + p_{i-1}$	0
	$x_i + y_i + p_{i-1} \geq k$	$x_i + y_i - k$	1
Вычитание	$x_i - p_{i-1} \geq y_i$	$x_i - y_i - p_{i-1}$	0
	$x_i - p_{i-1} < y_i$	$k + x_i - y_i - p_{i-1}$	-1

5. Наибольшее быстродействие.

6. Простота формального аппарата для синтеза цифровых устройств.

Самая простая – двоичная СС; используется аппарат булевой алгебры.

7. Удобство взаимодействия пользователя с системой.

В соответствие с этими критериями может быть рассчитан комплексный показатель.

Максимальную оценку получит двоичная СС.

Используем ПОСС с естественным порядком весов.

x_i {0, 1}

Используемые весовые коэффициенты:

Таблица 1.10

i	2_i	i	2_i
0	1	**10**	1024
1	2	**11**	2048
2	4	**12**	4096
3	8	**13**	8192
4	16	**14**	16384
5	32	**15**	32768
6	64	**16**	65536
7	128	**17**	131072
8	256	**18**	262144
9	512	**19**	524288
		20	1048576

$$x_s\ x_{s-1}\ \dots\ x_1\ x_0\ x_{-1}\ x_{-2}\ \dots\ x_{-m}$$
$$2^s\ 2^{s-1}\ \dots\ 2^1\ 2^0\ 2^{-1}\ 2^{-2}\ \dots\ 2^{-m}$$

В КС широко используются двоично-кодированные СС:

- десятичная (BCD);
- восьмеричная (ОСТ);
- шестнадцатеричная (HEX).

В этих СС для представления каждой цифры исходной системы используется несколько двоичных разрядов: для восьмеричной – 3, десяти- и шестнадцатеричной – 4.

Группа из 3-х двоичных названий, называется **триадой**, а из 4-х – **тетрадой**.

Следует отметить, что восьме- и шестнадцатеричные системы относятся к особому классу систем с основанием 2^1, где 1 – натуральное число.

Это обуславливает особенности их широкого применения в процессе взаимодействия пользователя с КС.

С одной стороны, они облегчают восприятие двоичной информации, так как любое двоичное представление числа может быть разбито на триады или тэтрады, которые проще для визуального восприятия человеком.

С другой стороны, они часто используются в качестве промежуточных СС при переводе из десятичной в двоичную СС и наоборот.

Двоичная СС вводит понятие **основной структурной единицы информации**, которой является **бит**. Бит соответствует одному разряду числа, представленному в двоичной СС.

Однако бит является слишком мелкой единицей и не всегда удобен для практических приложений. Поэтому наряду с битом в КС получила распространение **машинная структурная единица информации** – **машинное слово**, под которым понимают упорядоченную совокупность бит, имеющую некоторый смысл, т.е. воспринимаемую основными устройствами машины как единое целое.

Длина машинного слова (МС) является одной из определяющих характеристик КС и выбирается, прежде всего, исходя из соображений точности. Таким образом, длина машинного слова является различной в КС различных классов.

Для унифицированного представления информации используют структурную машинно-независимую единицу – **байт**; совокупность из 8 смежных двоичных разрядов.

Длину МС обычно выбирают равной целому числу байтов.

Пример: 16 бит, 32 бит, 64 бит, 48 бит.

Байт удобен тем, что полностью покрывает всю символьную информацию и удобен для записи десятеричных и шестнадцатеричных тэтрад.

Практика

1.1 Перевод чисел из одной системы счисления в другую

Задание 1

Получить десятичный эквивалент для двоичных чисел.

$$101101_{(2)} = 1\cdot 2^0 + 0\cdot 2^1 + 1\cdot 2^2 + 0\cdot 2^3 + 0\cdot 2^4 + 1\cdot 2^5 = 1+4+8+32 = 45_{(10)}$$

Перевод числа из двоичной в десятичную систему счисления выполняется только представлением его в развёрнутой форме.

$$1001001_{(2)} = 1\cdot 2^0 + 1\cdot 2^3 + 1\cdot 2^6 = 1+8+64 = 73_{(10)}$$

$$1101101_{(2)} = 1\cdot 2^0 + 1\cdot 2^2 + 1\cdot 2^3 + 1\cdot 2^5 + 1\cdot 2^6 = 1+4+8+32+64 = 109_{(10)}$$

Варианты заданий для самостоятельного выполнения			
Вариант 1	**Вариант 2**	**Вариант 3**	**Вариант 4**
110111; 1100111	111001; 1001011	111100; 1010111	101110; 1110111

Задание 2

Преобразование двоичных чисел в восьмеричное и шестнадцатеричное представление.

$$010101101110_{(2)} = 2556_{(8)}$$

Для преобразования числа из двоичной СС в восьмеричную, необходимо разбить число на триады. Если последняя триада неполная, то её дополняют 0. Каждую триаду заменяют восьмеричной цифрой.

Для преобразования числа из двоичной СС в шестнадцатеричную, необходимо разбить число на тэтрады, и последнюю при необходимости дополнить 0.

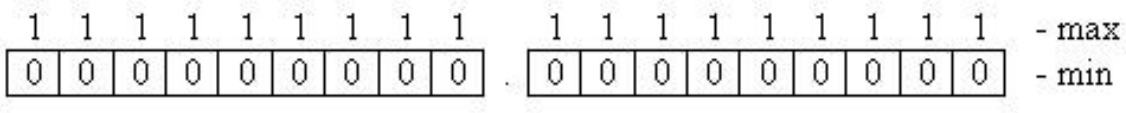
Если никаких мер не принимать, то цифрами шестнадцатеричной СС будут {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F}.

$$0001011110001110_{(2)} = 178E_{(16)}$$

Варианты заданий для самостоятельного выполнения			
Вариант 1	**Вариант 2**	**Вариант 3**	**Вариант 4**
010111011010	010110001110	001101111010	111011011010
0101011010001111	0111010111000110	1110101010001100	0111011011101010

Задание 3

Определить min и max число, которое можно записать в такую разрядную сетку.



$$a_8 2^8 + a_7 2^7 + \ldots + a_1 2^1 + a_0 2^0 + a_{-1} 2^{-1} + a_{-2} 2^{-2} + \ldots + a_{-9} 2^{-9}$$

Для системы с основанием k:

$$X_{\min} = 1 \cdot k^{-9}$$

$$X_{\max} = \sum_{i=-m}^{n} (k-1) \cdot k^i = k^{n+1} - X_{\min}$$

Количество чисел, которое можно записать в разрядную сетку: ?

Задание 4

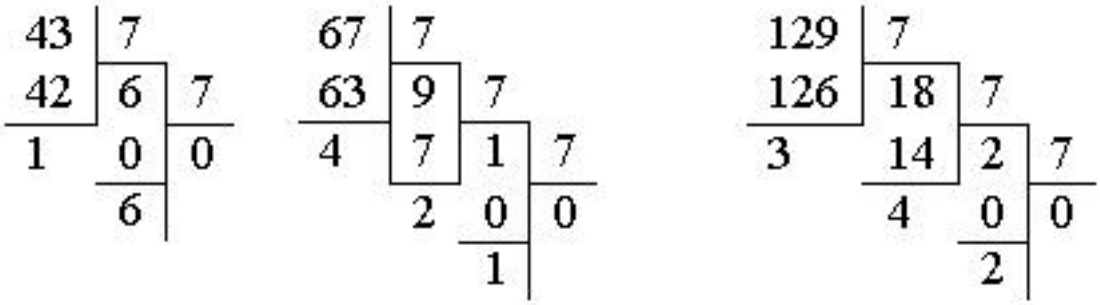
Перевести числа из десятеричной в семиричную СС.

Дано:

$43_{(10)} = 61_{(7)}$

$67_{(10)} = 124_{(7)}$

$129_{(10)} = 243_{(7)}$



Проверка: $243_{(7)} = 2 \cdot 7^2 + 4 \cdot 7^1 + 3 \cdot 7^0 = 2 \cdot 49 + 4 \cdot 7 + 3 = 2 \cdot 49 + 28 + 3 = 129_{(10)}$

Варианты заданий для самостоятельного выполнения			
Вариант 1	**Вариант 2**	**Вариант 3**	**Вариант 4**
3; 13	17; 5	24; 8	37; 7

Задание 5

Перевод неправильной дроби в двоичную, троичную и пятеричную СС.

Дано: $0,445_{(10)}$

0	445
	2
0	990
	2
1	780
	2
1	560
	2
1	120

0	445
	3
1	335
	3
1	005
	3
0	115
	3
0	045

0	445
	5
1	335
	5
1	005
	5
0	015

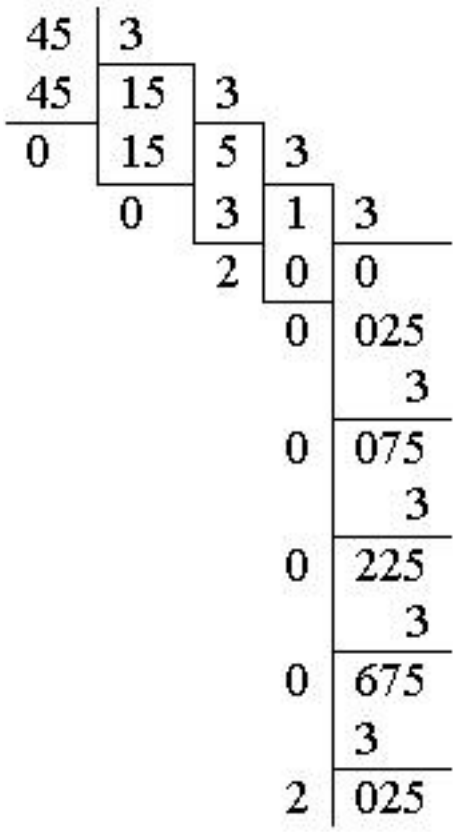
Ответ: $0,445_{(10)} = 0,0111_{(2)} = 0,011_{(3)} = 0,020103_{(5)}$

Варианты заданий для самостоятельного выполнения			
Вариант 1	**Вариант 2**	**Вариант 3**	**Вариант 4**
$0,515_{(10)}$	$0,236_{(10)}$	$0,410_{(10)}$	$0,358_{(10)}$

Задание 6

Перевод неправильных дробей в троичную СС.

Дано: $45,025_{(10)}$



Ответ: $45,025_{(10)} = 12.00.0002_{(3)}$

Варианты заданий для самостоятельного выполнения			
Вариант 1	**Вариант 2**	**Вариант 3**	**Вариант 4**
$23,014_{(10)}$	$57,135_{(10)}$	$48,576_{(10)}$	$35,259_{(10)}$

1.2 Способы представления чисел в ЭВМ. Дополнительный и обратный код

1	2
0000	1000
0001	1001
0010	1010
0011	1011
0100	1100
0101	1101
0110	1110
01111	1111

4-й разряд называется знаковым условно. Это есть признак поддиапазона.

Положительные числа в 1-м диапазоне, отрицательные - во 2-м диапазоне.

$$x_{np} = \begin{cases} x, x \geq 0 \\ A + |x|, x < 0 \end{cases}$$

$A = 2^{n-1}$

$A = 2^3 = 8$

Прямые коды содержат 2 нуля: 0.000 и 1.000

Для выполнения алгебраического сложения вводят коды:

- обратный код;
- дополнительный код.

Обратный код отрицательного числа есть дополнение его модуля до максимального числа, записанного в разрядную сетку.

$B = 2^{n-1} - 1$, n - знаковый разряд

$$|x| + |[x]_{обр}| = 2^{n-1} - 1.$$

Обратный код получают путём инвертирования разрядов модуля отрицательного числа.

Пример: -0110 = (пр)1.0110 => (обр)1.1001

Если слагаемое отрицательное, то находят обратный код, который участвует в операции.

x_1=0.1010=>10 дес

x_2=1.0111=>(обр)1.1000

$$\sum = |x_1| + |[x_2]_{обр}| = |x_1| - |x_2| + \underbrace{|x_2| + |[x_2]_{обр}|}_{1} = \underbrace{||x_1| - 1 - |x_2|| + 1}_{2} + |x_2| + |[x_2]_{обр}|$$

1 - это 2^{n-1} - 1 < 2^{n-1}.

2 - это 2^{n-1} (перенос в следующий знаковый разряд) - **циклический перенос.**

Значение будет отличаться от действительного на 1.

Если возникает циклический перенос, то алгебраическую сумму увеличиваем на 1.

Если перенос не возникает и операция алгебраической суммы представляется в обратном коде. Его превращают в прямой код.

x_1 = 0.1001

x_2 = 1.1110

$[x_2]_{обр}$ = 1.0001

0.1001

1.0001

1.1010 = -5

Дополнительный и обратный код

Задача 1

Получить обратный и дополнительный код, если изначально дано отрицательное десятичное число.

$-5;\ -37$

$-5_{(10)} = 1.101_{(2)} \rightarrow \text{ок} \rightarrow 0.010 \rightarrow \text{дк} \rightarrow 1.011$

$-37_{(10)} = 1.100101_{(2)} \rightarrow \text{ок} \rightarrow 0.011010 \rightarrow \text{дк} \rightarrow 1.011011$

Варианты заданий для самостоятельного выполнения			
Вариант 1	**Вариант 2**	**Вариант 3**	**Вариант 4**
$-9_{(10)}; -7_{(10)}$	$-13_{(10)}; -11_{(10)}$	$-3_{(10)}; -16_{(10)}$	$-5_{(10)}; -12_{(10)}$

Задача 2

Получить прямой код из обратного кода.

1.01110, 0.10101, 1.01101

1.01110 → ок → 1.10001

0.10101→ ок → 0.10101

1.01101 → ок → 1.10010

Варианты заданий для самостоятельного выполнения			
Вариант 1	**Вариант 2**	**Вариант 3**	**Вариант 4**
$1.01101_{(2)}$; $0.00111_{(2)}$	$0.01111_{(2)}$; $1.10011_{(2)}$	$0.10101_{(2)}$; $1.10111_{(2)}$	$0.11011_{(2)}$; $1.10001_{(2)}$

Контрольные вопросы

1. Для чего используют системы счисления отличные от десятичной?

2. Какие методы перевода чисел в СС с большим основанием существуют?

3. Какие методы перевода чисел в СС с меньшим основанием существуют?

4. Какие существуют способы представления чисел в ЭВМ?

5. Что такое мантисса числа?

6. В каком коде выполняются операции над знаковыми в ЭВМ?