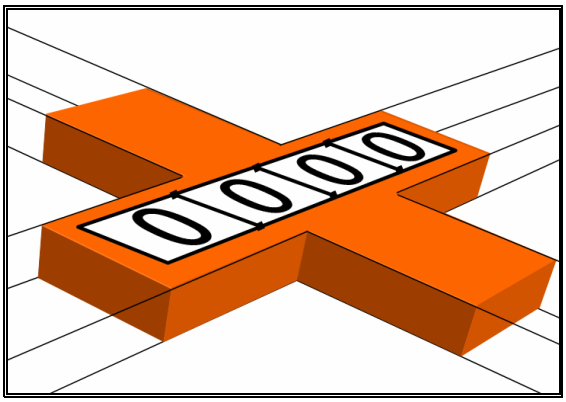
МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ АГЕНСТВО ПО ОБРАЗОВАНИЮ
ЯРОСЛАВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ им. П. Г. ДЕМИДОВА
математический факультет

И. В. Лаврентьев

СИСТЕМЫ СЧИСЛЕНИЯ. ВНУТРЕННЕЕ ПРЕДСТАВЛЕНИЕ ЧИСЕЛ В ЭВМ

ТЕОРИЯ И ЗАДАЧИ



ЯРОСЛАВЛЬ | 2008

УДК ***
ББК В**я**
Л **

Лаврентьев И. В. Системы счисления. Внутреннее представление чисел в ЭВМ. Теория и задачи. Учебно-методическое пособие / И. В. Лаврентьев; Яросл. гос. ун-т. – Ярославль: ЯрГУ, 2008. – 21 с.
ISBN *-****-****-*

Данное учебно-методическое пособие содержит следующие разделы дисциплины «Информатика»: системы счисления, внутреннее представление чисел в ЭВМ.

Предназначено для студентов первого курса университетов, обучающихся по специальности 010501 «Прикладная математика и информатика».

Пособие подготовлено с использованием текстового процессора Microsoft®Word® и редактора математических выражений MathType®.

Рис. 5. Библиогр. 5 назв.

ISBN *-****-****-*

Содержание

Раздел I. Системы счисления

Система счисления – способ представления чисел, основанный на определённом правиле выбора и записи символов, образующих число.

Системы счисления подразделяются на *позиционные* и *непозиционные*. Для позиционных систем счисления (ПСС) существенную роль играет положение каждой цифры от правого (или левого) края числа. Например, для десятичной системы счисления первый разряд слева от десятичной запятой обозначает количество единиц, второй – десятков и т.д. Справа от десятичной запятой идут по порядку десятые, сотые, тысячные, … доли числа.

Непозиционные системы счисления (НСС) не имеют указанного выше свойства, и положение цифр в них не играет в них роли значимости разряда. Примером непозиционной системы счисления может служить римская система.

§1.1. Непозиционные системы счисления

Как и было оговорено выше, в качестве примера непозиционной системы счисления можно рассмотреть римскую (РНСС). В РНСС существует четыре порядка цифр:

Порядок	**Цифры**
Единицы	**I** – единица, **V** = I+I+I+I+I
Десятки	**X** = V+V, **L** = X+X+X+X+X
Сотни	**C** = L+L, **D** = C+C+C+C+C
Тысячи	**M** = D+D

Натуральные числа записываются при помощи повторения этих цифр. При этом если большая цифра стоит перед меньшей, то они складываются (принцип сложения), если же меньшая — перед большей, то меньшая вычитается из большей (принцип вычитания). Последнее правило применяется только во избежание четырёхкратного повторения одной и той же цифры (например, вместо IIII пишут IV).

Пример. Пусть даны натуральные числа, записанные в десятичной системе счисления. Перевести их в римское представление:

а) 12; б) 49; в) 468; г) 512; д) 1003; е) 2008; ж) 5333

а) 12 = 10+2=X+II=XII
б) IL
в) CDLXVIII
г) DXII
д) MIII
е) MMVIII
ж) MMMMMCCCXXXIII

Римские числа особенны тем, что представление в них допускают лишь натуральные числа. Ещё одним неудобством является то, что такая форма записи чисел становится слишком громоздкой для натуральных чисел, больших третьего порядка, т.к. **M** здесь наибольшая из существующих значащих цифр.

Для перевода в римскую НСС натуральных чисел в десятичной арабской записи удобно использовать встроенную функцию пакета анализа MS Excel. Эта функция имеет прототип `=РИМСКОЕ(<число>)`. См. рисунок 1.

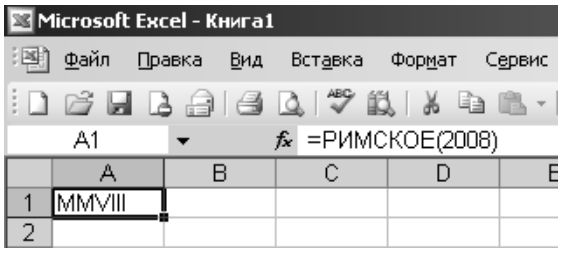


Рисунок 1. Перевод арабского числа в римское

Запись римских чисел чрезвычайно неудобна для арифметики. Например, сложно сразу определить каково значение выражения MMMMMCCCXXXIII – MIII. Однако, во многих случаях (в том числе, и в этом) можно воспользоваться правилом сложения и вычитания «столбиком»:

```
  MMMMMCCCXXXIII
–       M   III
-----------------
  MMMMCCCXXX
```

В качестве ещё одного примера непозиционной системы счисления можно привести и старославянскую систему записи чисел. Каждое из целых чисел от 1 до 9, а также десятки и сотни здесь обозначались буквами славянского алфавита с надписанным над ними знаком — ~ (титл).

Целые числа до 999 составлялись с помощью рядом стоящих славянских цифр. Например, $\overline{\text{ТКД}}$ = 324. Тысячи обозначались с помощью приставки к цифре, выражающей число тысяч некоторого знака.

Задачи.

1. Перевести в римское представление натуральные числа

а)	798	г)	1024	ж)	3210	к)	2999
б)	444	д)	2502	з)	1230	л)	4096
в)	666	е)	951	и)	3141	м)	2009

2. Перевести римские числа в десятичное арабское представление

а)	DXLI	ж)	CCXVII
б)	MMCCIII	з)	DCCLXXVII
в)	DCCCLXXXI	и)	DCXCVI
г)	CMLXXXIX	к)	CMLXIX
д)	MMVI	л)	MI
е)	DIV	м)	MMCCCXLV

3. Найти максимальное число, римская запись которого содержит ровно 20 цифр.

4. Существуют ли трёхзначные числа, в арабской записи которых совпадают первая и последняя цифры, а в римской записи числа есть росно 10 цифр?

5. Корректна ли римская запись числа $\underbrace{\text{MM...M}}_{1000}\underbrace{\text{CC...C}}_{100}\underbrace{\text{XX...X}}_{10}\text{I}$? Если нет, то предложить более корректную, найти арабский эквивалент числа.

§1.2. Позиционные системы счисления

Как уже и отмечалось ранее, главное отличие и достоинство позиционных систем счисления состоит в том, что разрядное положение влияет на её значимость. Например, для привычной нам десятичной системы счисления (с цифрами 0, …, 9) разрядность имеет следующий вид (таблица 1).

…	сотни	десятки	единицы	·	десятые	сотые	тысячные	…
	$d_2 \cdot 10^2$	$d_1 \cdot 10^1$	$d_0 \cdot 10^0$		$d_{-1} \cdot 10^{-1}$	$d_{-2} \cdot 10^{-2}$	$d_{-3} \cdot 10^{-3}$	

Таблица 1. Общий вид десятичного числа

То есть, любое конечное десятичное число однозначно представимо в виде $\overline{d_n \ldots d_0, d_{-1} d_{-2} \ldots}$, где $\overline{d_i}$ – цифры числа. Как видно из таблицы, числом «управляют» степени десятки. Отсюда напрашивается закономерный вопрос: «А что будет, если для числа выбрать другое “управляющее число”?». «Управляющее число» на самом деле называется *основанием системы счисления*. В общем случае для основания системы счисления можно выбирать любое натуральное число, тогда представление любого действительного числа W в произвольной p-ичной системе счисления ($p \in \mathrm{N}$) будет следующим:

$$W_p = w_n \cdot p^n + w_{n-1} \cdot p^{n-1} + \ldots + w_1 \cdot p^1 + w_0 \cdot p^0 + w_{-1} \cdot p^{-1} + w_{-2} \cdot p^{-2} + \ldots$$

В произвольной p-ичной ПСС возможно использовать ровно p цифр (символов, включая и тот символ, смысловая нагрузка которого ложится на ноль) для записи чисел. Наиболее распространенными ПСС являются двоичная (бинарная), восьмеричная, десятичная и шестнадцатеричная. Особо широко в вычислительной технике используются двоичная и шестнадцатеричная ПСС.

В качестве примера можно привести несколько разновидностей записи пяти чисел в пяти ПСС первого десятка.

ПСС				
унарная (p=1)	бинарная (p=2)	тернарная (p=3)	четверичная (p=4)	пятеричная (p=5)
–	0	0	0	0
1	1	1	1	1
11	10	2	2	2
111	11	10	3	3
1111	100	11	10	4

Таблица 2. Представление натуральных чисел 1, ..., 4 в различных ПСС

Унарная система счисления есть вырожденный случай ПСС при p=1; в ней невозможно подобрать представления для нуля.

§1.3. Двоичная ПСС. Перевод чисел [bin] ↔ [dec][1]

Двоичная (бинарная) ПСС одна из самых широко используемых систем счёта в компьютерной технике. Причиной этому послужил тот факт, что для записи любого числа в данной ПСС требуется всего два символа. Это чрезвычайно удобно для представления чисел в оперативной памяти и процессоре: нужно использовать всего лишь два дискретных состояния: «есть заряд» и «нет заряда».

Итак, бинарная ПСС имеет в качестве своего основания число 2 и алфавит цифр {0,1}. Общий вид числа в ней такой:

$$B = b_n \cdot 2^n + b_{n-1} \cdot 2^{n-1} + \ldots + b_1 \cdot 2^1 + b_0 \cdot 2^0 + b_{-1} \cdot 2^{-1} + b_{-2} \cdot 2^{-2} + \ldots$$

Суть бинарной системы заключается в разложении любого числа по степеням двойки

$$B = \sum_{i=m}^{n} b_i 2^i, \quad m \in \mathbf{Z}$$

Для перевода числа из десятичной системы в двоичную применяется следующий

Алгоритм

1. Выделить из числа целую и дробную части: $d = \mathrm{Trnc}(d) + \mathrm{Frct}(d)$;
2. Преобразовать целую часть выделением остатков при последовательном делении целой части десятичного числа на 2, пока очередное частное ≥2;
3. Преобразовать дробную часть выделением периода или нулей после запятой при последовательном умножении дробной части десятичного числа на 2.

Пример. Перевести число 383,125 из десятичной в двоичную ПСС. Переводим целую часть:

383	191	95	47	23	11	5	2	**1**
382	190	94	46	22	10	4	2	(1<2 ■) ↓
1	**1**	**1**	**1**	**1**	**1**	**1**	**0**	←

$383_{\text{dec}} = 101111111_{\text{bin}}$.

Переводим дробную часть:

0,	25
0,	50
1,	00 (нули ■)

$0{,}25_{\text{dec}} = 0{,}01_{\text{bin}}$.

Окончательно получаем: $383{,}125_{\text{dec}} = 101111111{,}01_{\text{bin}}$.

Теперь рассмотрим вопрос об *обратном переводе*. Чтобы осуществить его, нужно представить двоичное число как сумму произведений *i*-й его цифры на соответствующую степень двойки.

Пример. Перевести число $10010111001001{,}1110101001_{\text{bin}}$ в десятичную ПСС.

Имеем:

$$B_{bin} = \overset{13\,12\cdots}{1\,0\,0\,1\,0\,1\,1\,1\,0\,0\,1\,0\,0\,1}\overset{\cdots\,1\;0\;-1\,-2\cdots}{,\,1\,1\,1\,0\,1\,0\,1\,0\,0\,1}\overset{\cdots\,-10}{}_{bin} = 1\cdot 2^{13} + 1\cdot 2^{10} + 1\cdot 2^{8} + 1\cdot 2^{7} + 1\cdot 2^{6} + 1\cdot 2^{3} + 1\cdot 2^{0} +$$

$$+ 1\cdot 2^{-1} + 1\cdot 2^{-2} + 1\cdot 2^{-3} + 1\cdot 2^{-5} + 1\cdot 2^{-7} + 1\cdot 2^{-10} = 9673 + \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{32} + \frac{1}{128} + \frac{1}{1024} =$$

$$= (9673\frac{937}{1024})_{dec}$$

В двоичной ПСС точно так же, как и в десятичной, возможно производить простейшие арифметические операции, используя правила сложения и умножения в столбик. Для этого используются таблицы двух бинарных операций над множеством {0, 1}.

+	**0**	**1**
0	0	1
1	1	10

×	**0**	**1**
0	0	0
1	0	1

Пример. Произвести операции в арифметической форме

а) 100101010010 + 101010111010;

б) 111111110101 – 1001010010;

в) 1001 × 1011;

г) 10000 : 1000.

$$\begin{array}{r} 100101010010 \\ +\ 101010111010 \\ \hline 1010000001100 \end{array} \qquad \begin{array}{r} 111111110101 \\ -\ 1001010010 \\ \hline 110110100011 \end{array}$$

[1] Здесь и далее используются следующие обозначения:
bin – двоичная, **oct** – восьмеричная, **dec** – десятичная, **hex** – шестнадцатеричная ПСС.

```
      1001
×     1011
-----------
      1001
     1001
+   0000
   1001
-------
  1100011
```

```
 10000 |1000
-10000   10
    0
```

Следует отметить ещё одну особенность двоичных чисел. Поскольку основанием системы счисления является 2, то при правом приписывании n нулей к числу оно увеличивается в 2^n раз. И наоборот, если сдвинуть запятую целого числа на n позиций влево, то оно уменьшится в 2^n раз.

Пример. Обозначим сдвиг влево на n позиций двоичного числа b (запятая сдвигается при этом вправо, и число увеличивается) как SBAL(b, n) и аналогичный сдвиг вправо (запятая – влево, число уменьшается) – SBAR(b, n)[1]. Вычислить и получить десятичный ответ:

а) SBAR(<15_{10}>, 1)[2];
б) SBAL(101101011101, 4)
в) SBAR(SBAL(SBAR(<456768981231_{10}>, 2265), 2270), 5).

а) SBAR(1111, 1) = 111 = 7_{10}; б) SBAL(101101011101,4) = 1011010111010000 = 46544_{10};
в) 2265–2270+5 = 0 (сдвиги компенсируются). Результат 456768981231_{10}.

Задачи.

1. Перевести числа из десятичной системы счисления в двоичную:

а) 31
б) 1025
в) 146
г) 978
д) 15,0625
е) 625,25
ж) 233,(3)
з) 71,(71)

2. Перевести двоичные числа в десятичные:

а) 10110101110101
б) –1101010110
в) 1010,1111111
г) 11,0010010000111111011010

3. Выполнить указанные операции над бинарными величинами, не используя инженерный калькулятор

а) (10111+1010) div (10111–1111)
б) (11101·11011011+10001 div 11) div 110

4. Во сколько раз запись двоичных чисел длиннее записи десятичных?

5. Во сколько раз длиннее запись чисел произвольной p-ичной ПСС по сравнению другой q-ичной? ($p < q$)

6. Как можно определить по виду двоичной записи числа его чётность? Кратность 8?

7. Показать, что существует бесконечное множество двоичных чисел вида $\underset{2n}{10...0}$ для которых выполнено $\sqrt{\underset{2n}{10...0}} = \underset{n}{10...0}$, $n \in \mathrm{N}$.

8. При каких p и q выполнено $\underset{n}{10...0} = \sqrt[q]{(\underset{m}{10...0})^p}$? (m, n – фиксированы; все параметры натуральные).

§1.4. Восьмеричная ПСС. Перевод чисел по схемам [dec]↔[oct], [bin]↔[oct]

Восьмеричная ПСС имеет своим основанием натуральной число 8, т.е. в записи восьмеричных числе используются цифры 0, …, 7. Общий вид восьмеричного числа согласуется с общим видом числа в произвольной p-ичной ПСС и имеет вид:

$$C = c_n \cdot 8^n + c_{n-1} \cdot 8^{n-1} + ... + c_1 \cdot 8^1 + c_0 \cdot 8^0 + c_{-1} \cdot 8^{-1} + c_{-2} \cdot 8^{-2} + ...$$

Если сравнивать эту системы счисления с только что рассмотренной двоичной, то легко понять, что любое действительное число в данном случае разлагается в сумму по целым степеням восьмёрки с некоторыми коэффициентами – цифрами восьмеричного числа:

---

[1] SBAL – “Shift Binary Arithmetic operand Left”; SBAR – “Shift Binary Arithmetic operand Right”

[2] Если число записано под знаком операции в символах < · $_p$>, то это подразумевает первоначальный его перевод из p-ичной в допускаемую операцией ПСС (в данном случае двоичную), поскольку определённые выше операции допускают лишь бинарный первый аргумент.

$$C = \sum_{i=m}^{n} c_i 8^i, \quad m \in \mathbf{Z}$$

Пусть требуется перевести десятичное число в восьмеричное представление. Для этого целесообразно использовать

Алгоритм перевода десятичного числа в восьмеричное

1. Выделить из десятичного числа целую и дробную части;
2. Преобразовать целую часть выделением остатков при последовательном делении целой части десятичного числа на 8, пока очередное частное ≥8;
3. Преобразовать дробную часть выделением периода или нулей после запятой при последовательном умножении дробной части десятичного числа на 8.

Пример. Перевести десятичное число 888,17 в восьмеричную ПСС. Переводим целую часть:

888	111	13	**1**
888	104	8	(1<8 ■) ↓
0	**7**	**5**	←

$888_{dec} = 1570_{oct}$.

Переводим дробную часть:

0,	17
1,	36
2,	88
7,	04
0,	32
2,	56
4,	48
3,	84
6,	72
5,	76
6,	08
0,	64
5,	12
0,	96
7,	68
5,	44
3,	52
…	

В итоге получаем: $888{,}17_{dec} = 1570{,}1270243656050753\ldots_{oct}$.

Обратный перевод (из восьмеричной в десятичную ПСС) осуществляется таким же образом, как и из двоичной ПСС в десятичную, лишь с тем отличием, что суммирование осуществляется по степеням восьмёрки с нужными коэффициентами. Рассмотрим

Пример. Перевести число 701,46 в десятичную ПСС из восьмеричной.

$$C_{oct} = {}^{2\ 1\ 0\ -1-2}7\,01,4\,6 = 7\cdot 8^2 + 1\cdot 8^0 + 4\cdot 8^{-1} + 6\cdot 8^{-2} = 448 + 1 + \frac{1}{2} + \frac{3}{32} = (449\frac{19}{32})_{dec}.$$

Теперь рассмотрим вопрос о переводе числе из двоичной системы счисления в восьмеричную. В таком случае удобнее всего основываться на следующих рассуждениях. Основания двоичной и восьмеричной ПСС связаны кубической зависимостью. Следовательно, запись двоичного числа длиннее записи восьмеричного в $\log_2 8 = 3$ раза. Значит, на каждую восьмеричную цифру приходится некоторая триада двоичных цифр. Вот таблица таких триад:

0 ↔ 000	**2** ↔ 010	**4** ↔ 100	**6** ↔ 110
1 ↔ 001	**3** ↔ 011	**5** ↔ 101	**7** ↔ 111

Таблица 3. Соответствие триад двоичной ПСС цифрам восьмеричной

Пример. Перевести число 1570 из восьмеричной ПСС в бинарную.

$$\begin{matrix} 1 & 5 & 7 & 0 \\ 001 & 101 & 111 & 000 \end{matrix}, \quad m.e. 1101111000_{bin}$$

Обратный перевод осуществляется по этой же схеме.

Восьмеричная система, точно так же как и любая другая ПСС допускает выполнение арифметических действий над числами «в столбик». Для выполнения таких операций можно использовать восьмеричные таблицы сложения и умножения, приведённые ниже.

восьмеричная таблица сложения

+	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0	0	1	2	3	4	5	6	7
1	1	2	3	4	5	6	7	10
2	2	3	4	5	6	7	10	11
3	3	4	5	6	7	10	11	12
4	4	5	6	7	10	11	12	13
5	5	6	7	10	11	12	13	14
6	6	7	10	11	12	13	14	15
7	7	10	11	12	13	14	15	16

восьмеричная таблица умножения

×	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7
2	0	2	4	6	10	12	14	16
3	0	3	6	11	14	17	22	25
4	0	4	10	14	17	24	30	34
5	0	5	12	17	24	31	36	43
6	0	6	14	22	30	36	44	52
7	0	7	16	25	34	43	52	61

В восьмеричной система счисления сдвиги чисел осуществляются подобно бинарным, однако число изменяется уже в 8^n раз. Введём похожие операции восьмеричных сдвигов: SOAR(c, n) и SOAL(c, n).

Задачи.

1. Перевести в восьмеричную ПСС десятичные числа
а) 512,223
б) 66,(3)
в) 2,718281828459045

2. Перевести в десятичную ПСС из восьмеричной числа
а) 1234567
б) 777,7
в) 123,321

3. Выполнить операции над восьмеричными числами, не используя инженерный калькулятор:

а) $\mathrm{Trnc}\left(746+\frac{315}{17}\right)\mathrm{div}\left[\mathrm{Trnc}\left(75-\frac{144}{20}\right)\right]$

б) $40\cdot\mathrm{Trnc}\ \sqrt{221}\ -\mathrm{Frct}\left(4-\frac{13}{7}\right)$

в) $\mathrm{Trnc}\left(\frac{\mathrm{SOAL}(746.2,1)}{4}\right)$

4. Перевести числа из восьмеричной ПСС в двоичную:
а) 1515
б) 236
в) 765
г) 6666
д) 1010
е) 6274
ж) 1234567
з) 3571432
и) 7654321

5. Перевести числа из двоичной ПСС в восьмеричную:
а) 100101101011010011001
б) 11010111010100101
в) 11101011011101010
г) 101110100100100110011010 1
д) 10100111111111110101101
е) 100110110111011110101
ж) 101011110100010110000
з) 11111111100000001111

6. Вычислить в восьмеричной форме (матрицы содержат восьмеричные числа):

а) $\begin{pmatrix}1 & 3\\ 5 & 7\end{pmatrix}\cdot\begin{pmatrix}2 & 4\\ 10 & 20\end{pmatrix}$

б) $\begin{vmatrix}1 & 3 & 5\\ 7 & 10 & 1\\ 1 & 4 & 7\end{vmatrix}$

§1.5. Шестнадцатеричная ПСС. Перевод чисел по схемам [hex]↔[dec], [hex]↔[oct], [hex]↔[bin]

Шестнадцатеричная система счисления также является одной из наиболее используемых в вычислительной технике благодаря компактности представления чисел по сравнению, например, с десятичной или двоичной ПСС. Действительно, число, записанное в шестнадцатеричной системе счисления будет короче десятичного в $\log_{10}16$ раз и короче двоичного в $\log_2 16=4$ раза.

Представление любого числа в шестнадцатеричной ПСС строится на разложении его в сумму степеней числа 16 с коэффициентами – цифрами шестнадцатеричного числа. Так, справедливо следующее представление:

$$H=\sum_{i=m}^{n}h_i 16^i,\quad m\in\mathbf{Z}$$

Таким образом, любое число можно представить с шестнадцатеричном виде так:

$$H = h_n \cdot 16^n + h_{n-1} \cdot 16^{n-1} + ... + h_1 \cdot 16^1 + h_0 \cdot 16^0 + h_{-1} \cdot 16^{-1} + h_{-2} \cdot 16^{-2} + ...$$

При записи шестнадцатеричного числа используются 16 цифр: 0, 1, …, 9, A, B, C, D, E, F. Причём, A – эквивалент 10_{10}, B – эквивалент 11_{10}, …, F – эквивалент 15_{10}.

Приведём пример ряда 20 первых натуральных шестнадцатеричных чисел:

1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, 10, 11, 12, 13, 14.

Схема перевода десятичного числа в шестнадцатеричное стандартна и аналогична приводимым ранее схемам. Поэтому, рассмотрим сразу

Пример. Перевести число 249 из десятичного представления в шестнадцатеричное.

249	**F**
240	$(\blacksquare \le F)\downarrow$
9	←

$249_{dec} = F9_{hex}$.

Пример. Перевести число 7FFF из шестнадцатеричного представления в десятичное.

$$H_{hex} = 7FFF = 7 \cdot 16^3 + F \cdot 16^2 + F \cdot 16^1 + F \cdot 16^0 = 28672 + 3840 + 240 + 15 = 32767_{dec}$$

Рассмотрим подробнее вопрос о переводе в обе стороны восьмеричных и шестнадцатеричных чисел. Проследим связь длины из записи. Шестнадцатеричная запись длиннее восьмеричной в $\log_8 16 = \log_{2^3} 2^4 = \frac{4}{3}$ раза. То есть существует некая связь между восьмеричными группами цифр и шестнадцатеричными. Она такова, что на каждые четыре восьмеричные цифры приходится по три шестнадцатеричных:

0000 ↔ **000**	…	0317 ↔ **0FF**	…
0001 ↔ **001**	0020 ↔ **01F**	…	7776 ↔ **FFE**
0017 ↔ **00F**	…	7751 ↔ **FE9**	7777 ↔ **FFF**

Таблица 4. Соответствие тетрад восьмеричной системы триадам шестнадцатеричной

Проследив такую связь (её точную закономерность предлагается выяснить читателю), можно без труда перевести шестнадцатеричные числа восьмеричные, и наоборот.

Пример. Перевести 7FE9 из шестнадцатеричного представления в восьмеричное, 257 – из восьмеричного в шестнадцатеричное.

$$7FE9_{hex} = 007FE9_{hex} = 77751_{oct};\ 257_{oct} = 0257_{oct} = AF_{hex}.$$

Рассмотренная только что схема перевода используется крайне редко и не имеет, практически, никакого практического значения. Более интересен перевод из бинарной системы счисления в шестнадцатеричную, и обратно, так как именно эти системы счисления положены в основу полного и сокращённого (в смысле длины записи чисел) представления информации в ЭВМ.

Легко заметить, что двоичное число длиннее шестнадцатеричного ровно в $\log_2 16 = 4$ раза (этот факт отмечался в начале данного параграфа). Таким образом, мы можем связать тетрады двоичных чисел с шестнадцатеричными цифрами следующим образом:

0 ↔ 0000	**4** ↔ 0100	**8** ↔ 1000	**C** ↔ 1100
1 ↔ 0001	**5** ↔ 0101	**9** ↔ 1001	**D** ↔ 1101
2 ↔ 0010	**6** ↔ 0110	**A** ↔ 1010	**E** ↔ 1110
3 ↔ 0011	**7** ↔ 0111	**B** ↔ 1011	**F** ↔ 1111

Таблица 5. Соответствие между тетрадами двоичных чисел и шестнадцатеричными цифрами

Именно на указанном сейчас соответствии и строится взаимообратный перевод шестнадцатеричных и бинарных чисел.

Пример. Перевести число 7FD13CA в двоичное представление. Перевести число 1000101001101 в шестнадцатеричное представление.

$$7FD13CA_{hex} = 111111111010001001111001010_{bin}$$

$$0001000101001101_{bin} = 114D_{hex}.$$

Шестнадцатеричные числа можно складывать и умножать по обычным правилам действий с позиционными числами. Для таких действий существуют собственные шестнадцатеричные системы сложения и умножения (составить их предлагается читателю в качестве упражнения – см. задачу **1** после этого параграфа).

Заметим также, что над шестнадцатеричными числами определены операции шестнадцатеричных сдвигов (SHAR(*h*, *n*), SHAL(*h*, *n*)) вправо и влево (соответственно, уменьшение и увеличение в 16^n раз).

Задачи.

1. Составить шестнадцатеричные системы сложения и умножения без использования инженерного калькулятора.

2. Осуществить перевод десятичных чисел в шестнадцатеричное представление:

а) 102 г) 945,(16) ж) 23,75
б) 716 д) 1025,625 з) 63,125
в) 224,5 е) –17,99 и) –128,00390625

3. Перевести ответы предыдущей задачи из шестнадцатеричной системы счисления в десятичную в качестве проверки.

4. Перевести шестнадцатеричные числа в двоичные:

а) FD1C г) C69DA ж) DDEEFF
б) FF5AA д) ABCD з) 44CE5
в) 58E34 е) 7FFFF и) 9C1989

5. Перевести двоичные числа в шестнадцатеричные:

а) 101010100101111101 е) 11111111110111100001
б) 10101111101001010 ж) 110101100000111111101
в) 1100011110001110111011 з) 1111111101110111111101
г) 1001111111010110000001 и) 1110101001111011001010 1
д) 111001110011110111101 к) 1011110111110111111110111

6. Без использования инженерного калькулятора найти значение выражения в десятичной форме.

$$\left\{(7\mathrm{AF}_{hex}+140_{oct})\cdot\left[\left(\frac{3}{17}\right)_{dec}+\sqrt{79_{hex}}\right]+\mathrm{Frct}\left(\frac{1015_{oct}}{1111101_{bin}}\right)\right\}\cdot 19_{hex}.$$

Раздел II. Внутреннее представление чисел в ЭВМ

В этом разделе речь будет идти о представлении информации в ЭВМ. Поскольку вся информация кодируется цифрами и числами, то целесообразно будет понять, каким образом они представлены внутри компьютера (в оперативной памяти и процессоре). Материал этого раздела готовит читателя к знакомству с языком машинных команд низкого уровня – языком ассемблера.

§2.1. Архитектура ЭВМ. Типы данных. Зависимость представления чисел от конфигурации ЭВМ

В настоящее время существуют несколько подходов к построению (архитектуре вычислительных систем). Среди них, например, существуют модели систем последовательного и параллельного выполнения инструкций, систем реального времени и многие другие.

Особый интерес представляет стандартная модель организации вычислительной системы последовательного исполнения команд, предложенная Джоном фон Нейманом. Система состоит из центрального процессора, включающего в себя арифметико-логическое устройство (АЛУ), которое также называют сопроцессором, и непосредственно управляющую часть, которая координирует действия всей системы. Вычислительная система имеет конечную однородную и дискретную память с произвольным доступом (Random Access Memory – или просто RAM), состоящую из ячеек одинаковой размерности (8 бит – 1 байт). Доступ к данной памяти осуществляется дольше, чем к кэшу регистров сопроцессора, но быстрее, чет к постоянному запоминающему устройству (ПЗУ). Чаще всего роль ПЗУ играет жёсткий диск (Hard Disk Drive). Помимо постоянного запоминающего устройства имеются и так называемые устройства со съёмными носителями – дисководы Floppy, CD/DVD оптические приводы, накопители на флеш-платах USB (Universal Serial Bus) и портативных USB-HDD. Помимо перечисленных внутренних устройств вычислительной системы, имеются ещё и периферийные устройства – к ним можно отнести устройства ввода-вывода, такие как клавиатура, мышь, джойстик, Touchpad, принтер, сканер, модем, устройства воспроизведения и снятия звука и т.д. Внешние устройства, как правило, связываются с внутренней архитектурой либо посредством последовательных портов, либо с помощью концентраторов USB-портов. Последовательные порты, концентраторы, устройства внутренней периферии связаны с управляющей частью процессора, АЛУ и RAM через системную шину (системную магистраль). У различных типов платформ эта магистраль имеет различную разрядность. Для примера, в шестнадцатиразрядной операционной системе MS-DOS для платформы IBM PC магистраль была двадцатиразрядной, то есть имела разрядность 20 бит. Архитектуру фон Неймана можно условно представить следующей схемой (см. рисунок 2).

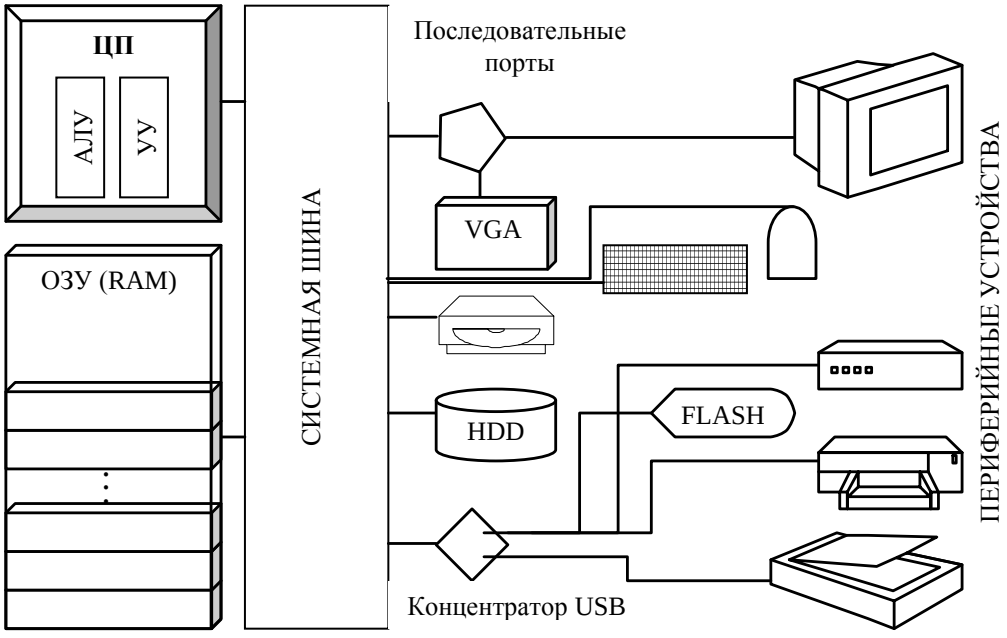


Рисунок 2. Архитектура ЭВМ по фон Нейману

Данные в любой вычислительной машине представимы как комбинация нулей и единиц, так как и регистры процессора и матрицы оперативной памяти энергозависимы и могут содержать или не содержать заряд на миркоконденсаторах.

Те, кто изучал хотя бы один язык программирования (например, Паскаль или C/C++), то, вероятно, знает, что данные в программе могут быть представлены различными типами, однако все они сводятся к цифровому представлению через кодирование определённым образом. Приведём список основных типов данных языков Паскаль и C.

Паскаль	**C/C++**
логический тип – *boolean*	логический тип – *bool*
символьный тип – *char*	символьный тип – *char*
числовые типы – *word, integer, real*	числовые типы – (*long/short*) *int, float, double*
строка – *string*	строка– *char[ ]*

Таблица 6. Некоторые типы данных языков Паскаль и C

Как видно из этой таблицы, основные типы данных в языках C и Паскаль совпадают. Числа представляются в памяти ЭВМ закодированными по некоторым правилам, суть которых и раскрывается в этом разделе. Символьный тип данных кодирует терминальные (печатные и непечатные) символы, сопоставляя им числа натурального ряда, которые в свою очередь могут быть представлены уже в числовом кодовом эквиваленте. Символьные строки – это последовательности подряд идущих в оперативной памяти символов.

Джон фон Нейман впервые предложил принцип организации работы программы, который заключался в хранении данных программы в отдельном блоке памяти для их возможных изменения и модификации в ходе работы программы. По этому принципу центральное вычислительное устройство должно лишь производить вычисления, выбирая нужные данные из памяти, которую само и адресует.

Центральный процессор любого компьютера состоит из особых ячеек быстрой памяти, называемых *регистрами*. Среди регистров микропроцессора существуют те, посредством которых можно осуществлять арифметические операции, с помощью которых адресуются сегменты памяти; есть регистр флагов, в котором выставляются и опускаются флаги при наступлении определённых событий.

Приведём схему строения микропроцессора на примере Intel Pentium *X*.

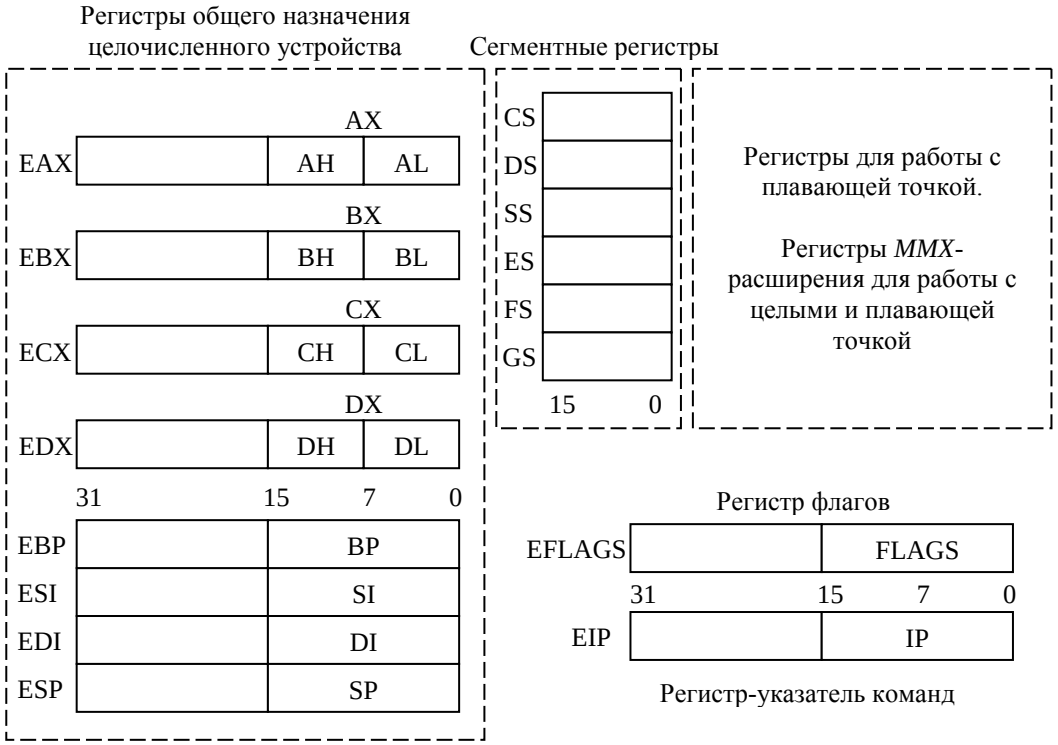


Рисунок 3. Регистры процессоров семейства i80x86

У каждого регистра процессора есть своё строго определённое назначение, однако многие из них могут использоваться для других целей. Например, регистр ECX предназначен для организации циклов, но это не мешает использовать его ещё и как простой арифметический регистр. Те, кто хоть немного знаком с языком *ассемблера*, поймут такую инструкцию процессору:

```
MOV ECX, 15AFH
ADD ECX, 23
```

Поясним, что означают команды в данной инструкции. Первая непосредственно заносит число 15AF (оно указано в 16-ной форме) в регистр ECX. Вторая команда прибавляет к текущему содержимому ECX число 23 (которое уже указано в десятичной форме).

Стоит отметить, что процессоры, которые были выпущены в недавнем времени, являются 32-разрядными, т.е. разрядность арифметических регистров в них составляет 32 бита. Однако вся серия процессоров i80x86 family построена таким образом, что каждая следующая модель полностью может эмулировать работу всех предыдущих. Поэтому при переходе от разрядности 16 к разрядности 32 за всеми 32-разрядными регистрами было оставлено из 16-разрядное подмножество. Например, 32-разрядный регистр EBX унаследовал 16-разрядного «предка» BX.

Таким образом, зависимость представления чисел в памяти и процессоре зависит от разрядности вычислительной машины, т.е., другими словами, от её поколения (давности выпуска).

Следующие параграфы будут посвящены непосредственно правилам представления различного типа чисел в памяти и процессоре.

§2.2. Внутреннее представление целых чисел

Целые числа – одни из самых используемых при решении многих задач на компьютере объекты. К примеру, каждый, наверное, писал строки наподобие «`var i,j: integer;`» на Паскале или «`int i,j;`» на С. Именно эти объявления переменных резервировали в памяти место, необходимое для размещения двух описанных переменных как двух целых величин. Условимся далее в этом параграфе рассматривать вычислительную машину с шестнадцатиразрядным процессором (или, по крайней мере, 32-раздарную машину в режиме эмуляции 16-разрядной подсистемы) для простоты изложения.

Мы оговаривали ранее, что память компьютера однородна, дискретная и разбита на ячейки размером по 1 байту. Для представления байта необходимо 8 битовых разрядов. То есть для записи в память целого числа в шестнадцатиразрядной подсистеме потребуется две ячейки памяти, идущие друг за другом и образующие *дуплет* для целого числа.

Каждый бит может кодировать два состояния – 0 или 1, т.е. используется двоичная система представления чисел. Определим, сколько целых чисел можно закодировать в 16 разрядах. Имеем $2^{16} = 65536$ различных чисел от 0000000000000000 до 1111111111111111.

Теперь определим следующие общепринятые понятия:

Определение 1. Прямым кодом целого числа называется его двоичное представление, дополненное нулями слева до конца разрядной сетки.

Определение 2. Инверсией двоичного кода называется инвертирование каждого бита в разрядной сетке (0→1, 1→0).

Определение 3. Обратным кодом целого числа называется его инверсный код плюс 1.

Теперь определим схему перевода целого числа (знакового или беззнакового):

Алгоритм

1. Число отрицательное? Да – перейти на шаг 3.
2. Перевести число в двоичное представление, выровнять его по правому краю разрядной сетки и заполнить нулями весь остаток левого края сетки. Перейти на шаг 6.
3. Перевести *модуль числа* по правилу шага 2.
4. Инвертировать разрядную сетку.
5. Получить обратный код в разрядной сетке.
6. Код числа получен. Конец алгоритма.

Понятно, что в разрядной сетке могут поместиться все числа от 16 нулей до 16 единиц. И тут возникает некий дуализм. Дело в том, что числа, представленные в обратном коде, ведут себя точно так же, как и отрицательные. И на самом деле для ЭВМ число из 16 единиц может рассматриваться и как максимальное беззнаковое число (65535), которое может быть представлено в рассматриваемой разрядной сетке, и как число «–1» – уже знаковое. Такой дуализм деления на знаковые и беззнаковые числа на самом деле безразличен для машины – это всего лишь условность, которую ввёл человек. Попытаемся привести множества чисел, которые можно представить в шестнадцатиразрядной сетке.

Множество целых беззнаковых чисел таково: $(\mathbb{N} \cup \{0\}) \cap [0; 65535]$. Множество целых знаковых составляют числа $\mathbf{Z} \cap [-32768; 32767]$. Действительно, начиная с 1000000000000000, знаковые числа становятся отрицательными и убывают *по абсолютной величине до единицы* при записи в разрядной сетке 16 единиц (число «–1»). Переводя число 1000000000000000 из двоичной ПСС в десятичную, имеем значение равное 32768 и вспоминаем про знак «–».

В качестве ещё одной иллюстрации двойственности знаковых и беззнаковых целых чисел читателю предлагается объяснить действие функции вывода стандартной библиотеки (`<stdio.h>`) языка С++ в следующих случаях (предполагается использование 16-разрядного компилятора):

1. `printf("%u",-15);` `//на выходе 65521`
2. `printf("%d",-15);` `//на выходе -15`

При записи целых чисел во внутреннем представлении, т.е. в двоичной форме, возникает одно неудобство – числа получаются слишком длинные. Для того, чтобы сделать их запись более компактной, используют перевод двоичного числа в шестнадцатеричное, и записывают его уже в шестнадцатеричной форме (см. §1.5.).

Пример. Перевести числа во внутреннее машинное представление, используя 16-разрядную сетку:

а) 15468
б) –3492

а) Число положительное. $15468_{dec} = 11110001101100_{bin}$. Имеем: 0011110001101100 или **3C6C**.

б) Число отрицательное. $3492_{dec} = 110110100100_{bin}$. Имеем: 0000110110100100. Инвертируем: 1111001001011011. Прибавляем 1: 1111001001011100 или **F25C**.

Представление в машинном коде целых чисел также иногда называют форматом с фиксированной точкой (ФТ).

Задачи.

1. Привести в формат ФТ следующие целые числа, используя шестнадцатиразрядную регистровую сетку:

а) 1989
б) –5123
в) 547
г) –2456
д) 9999
е) –545
ж) 2012
з) –2008

2. Привести в формат ФТ следующие пары целых чисел и сложить их по правилам бинарной арифметики:

а) ±512
б) ±4156
в) ±9478
г) ±32764
д) ±6789
е) ±3652

3. Описать алгоритм перевода чисел из формата ФТ в десятичное знаковое представление.

4. Пусть задан некоторый *язык низкого уровня*[1], в котором можно использовать одну шестнадцатиразрядную сетку `RG` (зарезервированное символическое имя), четыре арифметические операции:

- `ADD <симв_имя | непоср_число>;`
- `SUB <симв_имя | непоср_число>;`
- `MUL <симв_имя | непоср_число>;`
- `DIV <симв_имя | непоср_число>`, которые соответственно прибавляют к `RG`, вычитают из `RG`, умножают и целочисленно делят `RG` на заданное число или символическое имя переменной в памяти. Пусть также определены команды
- `MOV <симв_имя | непоср_число>`, которая помещает в `RG` заданное число или число по символическому имени, команда
- `NEW <симв_имя>`, создающая новую область в памяти для числа, и команда
- `LET <симв_имя>, <непоср_число | RG>` – оператор присваивания заданной переменной непосредственного числового значения или существующего значения регистра `RG`. Учитывая то, что данный язык распознаёт только приведённые к ФТ целые знаковые числа, описать формальные алгоритмы целочисленного вычисления следующих выражений:

а) `(((59-5)/2-7)/(-5))*18`
б) `(225/5-40)*20/(-100)`

§2.3. Внутреннее представление чисел с плавающей точкой

Помимо класса целых чисел выделяют класс вещественных – также представимых с определённой точностью в памяти компьютера. Таким образом, класс целых машинных чисел – лишь подмножество класса машинных чисел с плавающей точкой: $\mathrm{I} \subset \mathrm{R}$.

Определение 1. Стандартным видом десятичного вещественного числа $x \in \mathrm{R}$ называется такой его вид, что он удовлетворяет соотношениям:

$$x = M \cdot 10^P, \quad M \in [0;10) \cap \mathrm{R}, \quad P \in \mathbf{Z}. \qquad (*)$$

Составляющая *M* числа *x* называется мантиссой, число *P* называется порядком. Для примера рассмотрим пример перевода некоторых вещественных чисел в стандартный вид.

Пример. Перевести действительные числа в стандартный вид:

[1] Язык вымышлен и является в некотором смысле прототипом языка ассемблера.

а) 3,14159… = $3{,}14159\cdot10^{0}$; б) 206264,8 = $2{,}06264\cdot10^{5}$; в) 0,00002415 = $2{,}415\cdot10^{-5}$.

С понятием стандартного представления числа связано его экспоненциальное представление.

Определение 2. Экспоненциальным представлением числа называется его стандартное представление, удовлетворяющее соотношениям (*), записанное в виде

$$M\,\mathbf{E}+P\,.$$

Символ **E** указывает на то, что данное представление числа является экспоненциальным. Таким образом, становится ясно, что для хранения числа с плавающей точкой в памяти компьютера достаточно запомнить две его характеристики – вещественную мантиссу и целочисленный знаковый порядок.

Рассмотрим теперь вопрос о внутреннем представлении числа с плавающей точкой. Для этого введём в рассмотрение 32-х разрядную сетку, которую разобьём в соответствии с международным стандартом **IEEE** на три поля. Первое поле содержит один 31-й бит и будет называться знаком числа (*s*), второе поле, размещённое в битах с номерами 30, …, 23, будет называться смещённым порядком. Третье поле из оставшихся 23 битов с номерами 22, …, 0 отведём под мантиссу (см. рисунок 4).

s	смещённый порядок								мантисса																					
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00

Рисунок 4. Разрядная сетка для представления числа с плавающей точкой

Теперь опишем

Алгоритм перевода числа с плавающей точкой во внутреннее представление по стандарту **IEEE** для 32-х разрядной сетки.

1. Переводимое число положительно? Да – переход на шаг 2. Иначе – переход на шаг 3.
2. Выставить в 31-й разряд **0**. Переход на шаг 4.
3. Выставить в 31-й разряд **1**.
4. Перевести десятичную дробь (число) в двоичную ПСС.
5. Привести двоичную дробь к стандартному виду[1] и определить из него двоичный порядок P_2.
6. Вычислить смещённый двоичный порядок P_2' как $127_{10}+P_2$. Записать его в разряды с 30 по 23.
7. В стандартном представлении, полученном на шаге 5, ведущая и единственная целочисленная единица называется неявной – поскольку не несёт никакой дополнительной информации. Отбросить неявную единицу. Вся дробная часть (учитывая её ведущие нули при их наличии) является мантиссой, которая записывается с биты с номерами с 22 по 00.
8. Конец работы алгоритма.

Языки программирования, например, C++ (16-разрядной реализации), дают рамки представления чисел в формате 32-разрядной сетки (для C++ – это формат float) в рамках от 3,4E–38 до 3,4E+38.

В качестве примера переведём пару вещественных[2] чисел.

Пример. Получить машинное представление в 32-х разрядной сетке: а) 350,001; б) –248,3358.

а) $350{,}001_{dec} = 101011110{,}000000000100001_{bin}$. Приводим к стандартному виду полученное двоичное число: $1{,}01011110000000000100001\cdot(10_2)^8$. $P_2 = 8_{dec}$. $P_2' = (127+8)_{dec} = 10000111_{bin}$. Мантисса $M_2 = 01011110000000000100001$. Знак числа положителен, поэтому $s = 0$. Окончательно получаем: 01000011101011110000000000100001 или **43AF0021** (см. рисунок 5).

s	смещённый порядок								мантисса																					
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
0	1	0	0	0	0	1	1	1	0	1	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1

Рисунок 5. Представление числа 350,001 в формате ПТ

б) совершенно аналогично получаем представление **C37855F7**.

Для тех, кто уже хотя бы немного знаком с языком ассемблера, автор считает нужным привести ещё один верный способ перевода чисел в формат ПТ при помощи средств данного языка (в силу

[1] Стандартный вид числа в двоичном представлении таков: $1,\beta_k\beta_{k-1}\beta_{k-2}\ldots\cdot(10_2)^{P_2}$

[2] Слово «вещественное» здесь употреблено в не совсем верном смысле, так как в силу ограниченности и дискретности разрядных сеток в них могут быть представлены лишь рациональные числа (с точностью приближения к реальным числам из множества $\mathbb{R}\setminus\mathbb{Q}$ с некоторой погрешностью)

трудоёмкости такого перевода иногда бывает полезно проверить ручные расчёты с помощью вычислительной техники).

Целесообразно использовать, например, шестнадцатиразрядный компилятор языка ассемблера `TASM.EXE` (Turbo Assembler) от Borland и любой файловый менеджер (SC, TC, NC, DN, FAR или др.) либо средства командной строки.

В директории с файлом компилятора и другими системными файлами Турбо-ассемблера нужно создать файл с произвольным именем и расширением `*.asm` (например, `X.asm`). В нём с помощью любого простейшего текстового редактора забить следующие строки с кодом, описывающим сегмент данных с числами, которые необходимо перевести во внутреннее представление:

```
DATA SEGMENT
F1        DD        350.001
F2        DD        -248.3358
DATA ENDS
END
```

Далее, этот файл нужно сохранить и запустить на компиляцию следующей командой:
`TASM.EXE /ZI X.ASM,,,`

В результате, в текущей директории будет создан файл листинга, содержащий всю необходимую информацию. Ниже приведён фрагмент файла `X.LST`, полученного в результате вышеописанных действий. По существу его содержимое в комментариях не нуждается (по крайней мене в них не нуждается интересующая нас часть с полученным внутренним представлением чисел.)

```
Turbo Assembler  Version 3.1        18/06/08 00:32:31       Page 1
X.asm



      1 0000                        data segment
      2 0000  43AF0021              F1      dd      350.001
      3 0004  C37855F7              F2      dd      -248.3358
      4 0008                        data ends
      5                             end
…
```

Похожим образом можно переводить во внутренне представление и целые числа, однако эту сторону вопроса автор не затрагивает и оставляет её читателю для самостоятельного знакомства с языком ассемблера.

Задачи.

1. Представить числа в стандартном и экспоненциальном виде.

а) 512,146
б) 0,0000018
в) 18000000,01
г) –56,00001546
д) –24523,698491
е) 81,24569

2. Перевести во внутреннее представление формата ПТ32 следующие числа без использования вычислительной техники:

а) 2265,236
б) –912,1989
в) 3,14159
г) –2,71828
д) 479,0059
е) –15145,998
ж) 253,7354
з) –647,5984
и) 2256,4719
к) –986,4521
л) 1024,3984
м) –333,4456
н) 0,00018
о) –0,00051
п) 0,1234
р) –16,4742

3. Описать алгоритм обратного перевода чисел из внутреннего представления ПТ32 в десятичное и перевести следующие числа:

а) 43194189
б) C7C35000
в) 3280D959
г) B22BCC77

4. Проследив закономерности построения представления числа в 32-х разрядной регистровой сетке, построить представление следующих чисел в 64-х разрядной:

а) 35,1265
б) –9981,001

ОТВЕТЫ

РАЗДЕЛ I. СИСТЕМЫ СЧИСЛЕНИЯ

§1.1. **1.** DCCXCVIII, CDXLIV, DCLXVI, MXXIV, MMDII, CMLI, MMMCCX, MCCXXX, MMMCXLI, MMCMXCIX, MMMMXCVI, MMIX. **2.** 541, 2203, 881, 989, 2006, 504, 217, 777, 696, 969, 1001, 2345. **3.** 20000. **4.** ДА. Например, 383 = CCCLXXXIII. **5.** $\underbrace{\text{MM...M}}_{1010}\text{CI} = 1010101$.

§1.3. **1.** 11111; 10000000001; 10010010; 1111010010; 1111,0001; 1001110001,01; 11101001,(01); 1000111,10110111100110001001… **2.** 11637; –854; 10,9921875; 3,14159265358979. **3.** 100; 10000100011. **4.** $\log_2 10$. **5.** $\log_p q$. **6.** Целое двоичное число чётно тогда и только тогда, когда последняя цифра в его записи есть 0. Кратно восьми – тогда и только тогда, когда последние три цифры числа – нули.

§1.4. **1.** 1000,1621320712601014223351; 102,(25); 2,5576052130505355062724. **2.** 342391; 511,875; 83408203125. **3.** 11; $577\frac{3}{7}$; 1714. **4.** 1101001101; 10011110; 111110101; 110110110110; 1000001000; 110010111100; 1010011100101110111; 10101111001100011010; 111110101100011010001. **5.** 4553231; 327245; 353352; 135111465; 24777655; 4667365; 5364260; 3774017. **6.** $\begin{pmatrix} 32 & 64 \\ 102 & 204 \end{pmatrix}$; 10.

§1.5. **2.** 66; 2CC; E0,8; 3B1,295FAD... ; 401,A; –11,FD70A; 17,C; 3F,2; 80,01. **4.** 1111110100011100; 11111111010110101010; 1011000111000110100; 1100011010011101010; 1010101111001101; 1111111111111111111; 11011101111010111111111; 1000100110011100101; 10011100000110011000 1001. **5.** 2A97D; 15F4A; 31E3BBl; 27F581; 1CE7BD; FFDE1; 1AC1FD; 3FDDFD; 753D95; 5EFBF7. **6.** $\frac{4126000}{7}(dec)$.

РАЗДЕЛ II. ВНУТРЕННЕЕ ПРЕДСТАВЛЕНИЕ ЧИСЕЛ В ЭВМ

§2.2. **1.** 07C5; EBFD; 0223; F668; 270F; FDDF; 07DC; F828. **2.** числа в сумме при правильном сложении дают 0 для 16-ти разрядов (для 17-ти разрядов всегда получается 2^{16}).
4. для задачи «а»)

```
NEW RES                    ; комментарии
LET RES, 0000000000000000 ; RES=0
LET RG,  0000000000111011 ; RG=59
SUB      0000000000000101 ; RG=RG-5
DIV      0000000000000010 ; RG=RG/2
SUB      0000000000000111 ; RG=RG-7
DIV      1111111111111011 ; RG=RG/(-5)
MUL      0000000000010010 ; RG=RG*18
LET RES, RG                ; RES=RG
```

§2.3. **2.** 450D93C7; C4640CBB; 40490FD0; C02DF84D; 43EF80C1; C66CA7FE; 437DBC43; C421E64C; 450D078D; C4769CEF; 44800CC0; C3A6B909; 393CBE62; 393CBE62; BA05B185; 3DFCB924; C183CB29. **3.** 153,256; –99999,99999; 0,000000015; –0,00000001. **4.** 40048C8189374BC6A7EF; C00C9BF4010624DD2F1A.

Тематика задач для программирования

1. Взаимный перевод арабских и римских чисел.
2. Взаимный перевод чисел десятичной и двоичной систем счисления
3. Взаимный перевод чисел десятичной и восьмеричной систем счисления
4. Взаимный перевод чисел десятичной и шестнадцатеричной систем счисления
5. Взаимный перевод чисел восьмеричной и двоичной систем счисления
6. Взаимный перевод чисел восьмеричной и шестнадцатеричной систем счисления
7. Взаимный перевод чисел двоичной и шестнадцатеричной систем счисления
8. Автоматическое построение таблиц сложения и умножения для произвольной p-ичной ПСС ($p \geq 2$).
9. Реализация взаимного перевода чисел произвольной p-ичной ПСС и десятичной ПСС ($p \geq 2$).
10. Перевод целого числа в формат с фиксированной точкой для разрядных сеток 8, 16, 32, 64 бита.
11. Перевод действительного числа в формат с плавающей точкой для разрядных сеток 32 и 64 бита.
12. Моделирование калькулятора для осуществления расчётов в одной из систем счисления.
13. Моделирование калькулятора для осуществления расчётов с форматом чисел ФТ.
14. Моделирование представления «длинных целых» чисел и действий с ними при помощи векторов.

Список литературы

[1]. *Абель П.* Язык ассемблера для IBM PC и программирования. – М.: Высш. шк., 1992. – 447с., ил.
[2]. *Нортон П.* Персональный компьютер фирмы IBM и операционная система MS-DOS /Пер. с англ. – М.: Радио и связь, 1998. – 256 с.
[3]. *Орловский Г. В.* Введение в архитектуру микропроцессора 80386. – СПб.: BHV, 1997. В 2-х т.
[4]. *Финогенов К. Г.* Основы языка ассемблера. – М.: Радио и связь, 1999. – 288 с.
[5]. *Юров В.* Assembler: Учебник. – СПб. и др.: Питер, 2000. – 622 с.

Учебное издание

Лаврентьев Иван Викторович
Системы счисления. Внутреннее представление чисел в ЭВМ
Теория и задачи

Учебно-методическое пособие

Компьютерный набор и вёрстка И. В. Лаврентьев

Формат 60×84/8. Бумага DataCopy.
Гарнитура TTF Times
Тираж *** экз.