Министерство образования Республики Беларусь
Учреждение образования
«Белорусский государственный университет
информатики и радиоэлектроники»

Институт информационных технологий

Кафедра информационных систем и технологий

А. Г. Савенко, А. И. Парамонов

ОСНОВЫ КОМПЬЮТЕРНОЙ ТЕХНИКИ

Рекомендовано УМО по образованию в области информатики и радиоэлектроники в качестве учебно-методического пособия для специальностей
6-05-0611-05 «Компьютерная инженерия»,
6-05-0612-01 «Программная инженерия»

Минск БГУИР 2025

УДК 004.382(076)
ББК 32.971.321.4я73
С12

Р е ц е н з е н т ы:

кафедра информационных систем и технологий
учреждения образования
«Белорусский государственный технологический университет»
(протокол № 8 от 29.02.2024);

ректор учреждения дополнительного образования
«Институт информационных технологий и бизнес-администрирования»
кандидат технических наук, доцент В. К. Дюбков

Савенко, А. Г.

С12 Основы компьютерной техники : учеб.-метод. пособие / А. Г. Савенко, А. И. Парамонов. – Минск : БГУИР, 2025. – 132 с. : ил.
ISBN 978-985-543-783-4.

Предназначено для студентов первого курса специальностей «Программная инженерия» и «Компьютерная инженерия». Содержит теоретический материал, практические работы, а также примеры решения практических задач. Можно использовать для самостоятельной работы студентов указанных специальностей очной, заочной и дистанционной форм обучения. Будет полезно студентам всех специальностей профиля образования 06 – информационно-коммуникационные технологии и специалистам в области информационных технологий.

УДК 004.382(076)
ББК 32.971.321.4я73

ISBN 978-985-543-783-4

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Сегодня мы все чаще пользуемся достижениями человечества без понимания их принципов действия. Это допустимо для конечного потребителя продукта, но не для разработчика. Строитель, который закладывает крышу, безусловно должен понимать основы построения нижней части дома. Подобных примеров можно указать еще множество, но ограничимся утверждением: «Разработчику программного обеспечения любого уровня необходимо иметь понимание того, что и как работает на уровне представления компьютером». В настоящее время компьютерная техника внедряется во все сферы человеческой деятельности, а знания в этой области являются необходимыми для специалистов, разрабатывающих программное обеспечение, в том числе и программное обеспечение встроенных систем.

Данное пособие ставит целью ознакомить студентов с арифметическими основами компьютерной техники, помочь им приобрести теоретические знания и практические навыки выполнения основных арифметических операций.

Наряду с обширным теоретическим материалом учебно-методическое пособие включает рекомендации по выполнению практических работ, варианты заданий для практических работ, затрагивающие все основные темы указанных выше учебных программ. Приведены справочные материалы, примеры решения типовых задач и представлены ответы на практические задания для самопроверки студентов.

Учебно-методическое пособие в первую очередь ориентировано на студентов заочной формы получения высшего образования по программам, интегрированным с образовательными программами среднего специального образования, по специальностям 6-05-0611-05 «Компьютерная инженерия» и 6-05-0612-01 «Программная инженерия» и написано в соответствии с учебными программами учебных дисциплин «Основы компьютерной техники» и «Арифметические и логические основы цифровых устройств». Также оно может быть полезно студентам всех специальностей группы «Разработка программного обеспечения» и использовано для профессиональной ориентации в ходе подготовки к получению квалификации инженера-программиста.

ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

1 ПРЕДСТАВЛЕНИЕ ИНФОРМАЦИИ В ЭЛЕКТРОННЫХ ВЫЧИСЛИТЕЛЬНЫХ МАШИНАХ

1.1 Системы счисления

Представление и арифметическая обработка чисел во многом определяется используемой системой счисления, которая представляет собой совокупность допустимых цифр (символов) и набор правил, позволяющих однозначно представлять числовую информацию.

Система счисления – это определенный способ представления чисел, определяющий их символическую запись и соответствующие ему правила действий над числами, определяющие их алгебраическую и арифметическую суть.

Система счисления – совокупность приемов обозначения чисел; язык, алфавитом которого являются символы (цифры), а синтаксисом – правило, позволяющее сформулировать запись чисел однозначно.

В свою очередь система счисления характеризуется рядом других понятий, таких как:

1 *Цифра* – это символьное обозначения, используемое для записи чисел. В основном представляет собой один символ, т. е. минимальную, неделимую на составляющие часть в записи числа, и соответствует одному *разряду* числа.

Существуют различные наборы цифр: арабские (индо-арабские), римские (для записи используется латинский алфавит), цифры Майя и др.

Примеры цифр различных систем счисления: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, I, V, X, L, M, Δ, Г, Ω, א, Д, Ц и др.

2 *Число* – запись, состоящая из цифр и отражающая количественную оценку в соответствии с правилами конкретной системы счисления.

3 *Алфавит системы счисления* – это множество цифр в системе счисления, используемое для записи чисел.

4 *Код числа* – запись числа в некоторой системе счисления.

Классификация систем счисления

Общая классификация систем счисления представлена на рисунке 1.1.

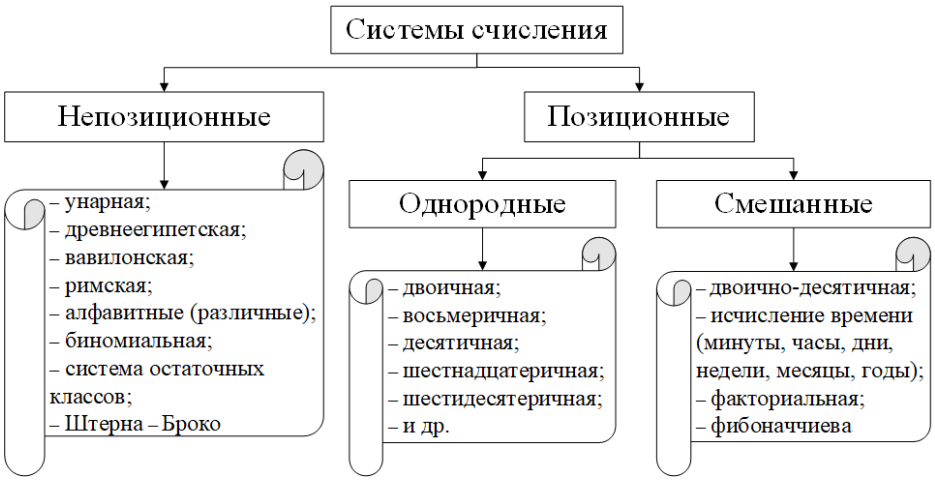


Рисунок 1.1 – Классификация систем счисления

Непозиционные системы счисления характеризуются тем, что доля («вес», количественный эквивалент) цифры не изменяется и не зависит от местоположения в записи числа, а количественная оценка числа определяется как сумма цифр в записи числа.

В общем случае количественную оценку числа, записанного в непозиционной системе счисления, можно представить как

$$\text{Количественная оценка} = \sum_{i=0}^{n-1} A_i, \qquad (1.1)$$

где n – количество разрядов (количество цифр) в записи числа;

i – номер разряда, в котором записана цифра;

A – цифра в записи числа.

Алфавит в непозиционной системе счисления содержит неограниченное количество символов.

К наиболее распространенным непозиционным системам счисления относятся:

1 *Унарная система счисления* – наиболее архаичная система счисления, в которой одна условная цифра обозначает одну единицу счета. В качестве цифр использовались узлы, засечки, камни и т. д. Примеры использования унарной системы счисления представлены на рисунке 1.2.

Рисунок 1.2 – Примеры применения унарной системы счисления

2 *Египетская система счисления* применялась в Древнем Египте вплоть до начала X в. н. э. В этой системе цифрами являлись иероглифические символы, которые обозначали числа, кратные степеням десяти: 1 (палка), 10 (ярмо), 100 (веревка), 1000 (лотос), 10 000 (палец), 100 000 (лягушка), 1 000 000 (поклонение), 10 000 000 (Ра) (рисунок 1.3).

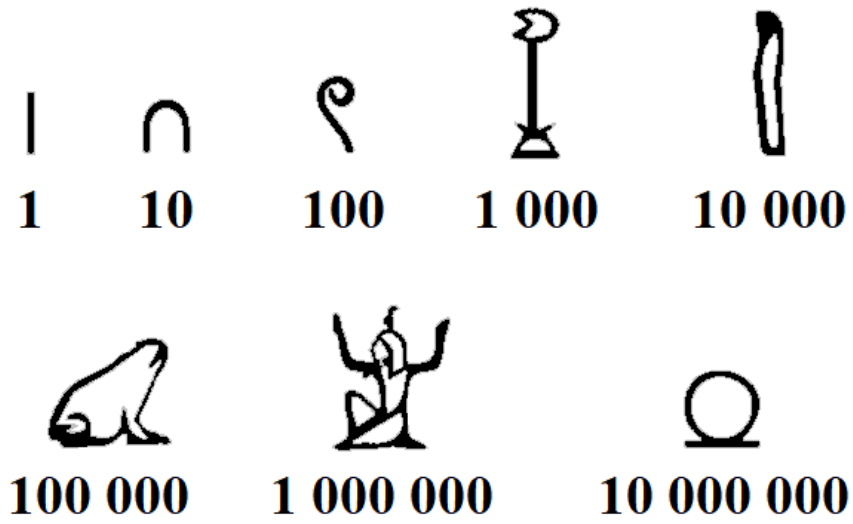


Рисунок 1.3 – Примеры чисел египетской системы счисления

3 *Римская система счисления* как наследие римской цивилизации используется по настоящее время для обозначения летоисчисления (столетий) от Рождества Христова практически во всем мире.

В римской системе счисления используются следующие обозначения цифр, используемых для записи чисел:

– **I** – 1 (палец);
– **V** – 5 (раскрытая ладонь, 5 пальцев);
– **X** – 10 (две ладони);
– **L** – 50;
– **C** – 100 (Centum);
– **D** – 500 (Demimille);
– **M** – 1000 (Mille).

Примеры обозначения чисел в римской системе счисления представлены на рисунке 1.4.

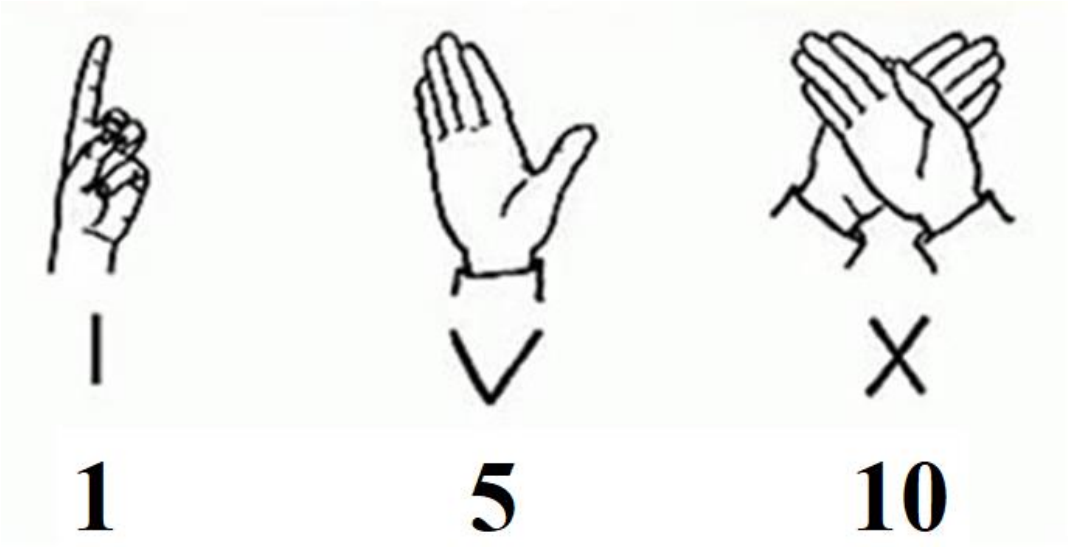


Рисунок 1.4 – Примеры обозначения чисел римской системы счисления

Примеры записи чисел в римской системе счисления представлены в таблице 1.1.

Таблица 1.1 – Примеры записи чисел в римской системе счисления (СС)

Запись в римской СС	Определение количественной оценки	Правила определения количественной оценки числа	Значение в десятичной СС
XXI	10 + 10 + 1	Каждая меньшая по значению цифра, записанная справа от большей по значению, прибавляется ней. Каждая меньшая по значению цифра, записанная слева от большей по значению, вычитается из нее	21
CXV	100 + 10 + 5		115
MIII	1000 + 1 + 1 + 1		1003
XIV	10 + (–1) + 5		14

4 *Алфавитная система счисления* ставит в соответствие каждой букве национальных алфавитов значение определенной цифры (или даже числа). Разновидностями алфавитной системы счисления являются греческая, еврейская, индийская, славянская и др. Пример славянской алфавитной системы счисления представлен на рисунке 1.5.

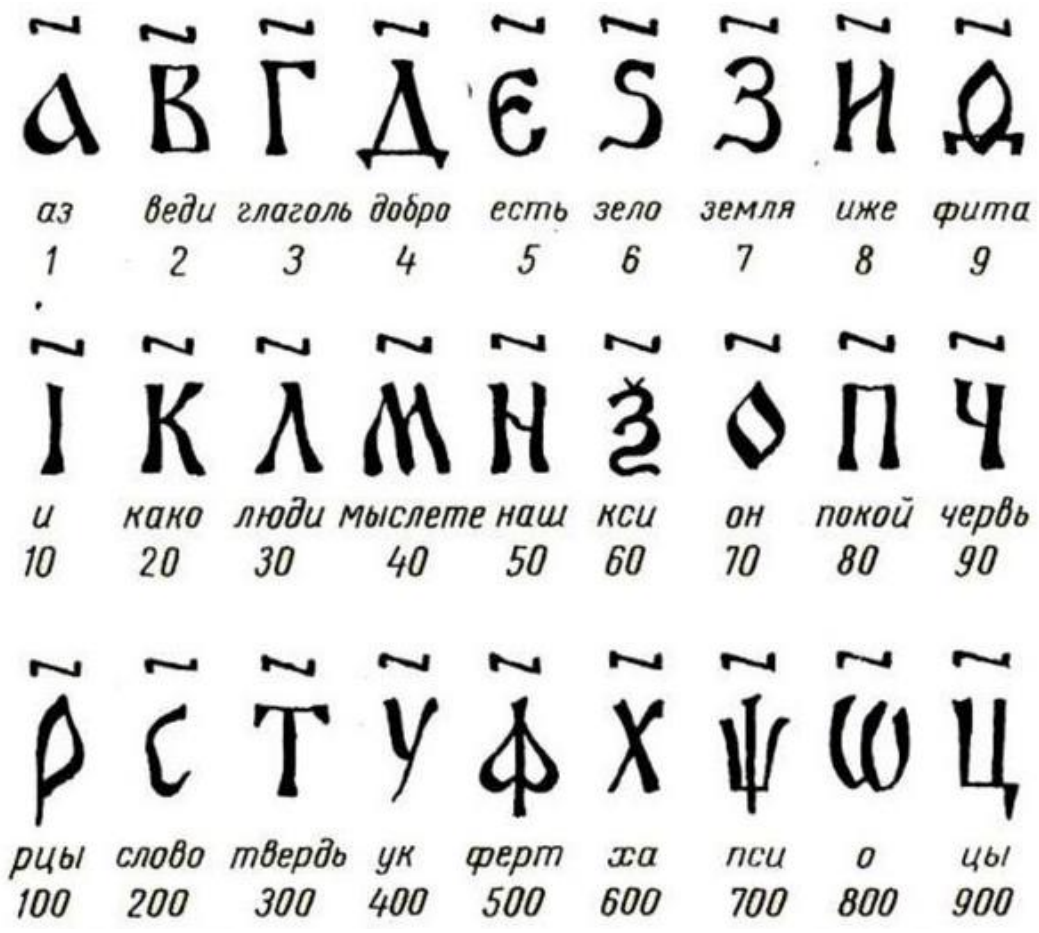


Рисунок 1.5 – Пример славянской алфавитной системы счисления

Позиционные системы счисления характеризуются тем, что доля («вес») некоторой цифры в количественной оценке записанного числа определяется не только видом цифры, но и местоположением (позицией) данной цифры в записи числа, т. е. каждая позиция (разряд) в записи числа имеет определенный «вес», а количественная оценка записанного числа в такой системе счисления определяется как сумма произведений значений цифр, составляющих запись числа, умноженных на вес позиции, в которой располагается цифра.

Алфавит в позиционной системе счисления ограничен заданным количество символов.

Также позиционные системы счисления характеризуются такими понятиями, как основание системы счисления, разряд, номер разряда, разрядность числа.

Основание системы счисления – это количество цифр в алфавите системы счисления (далее будет обозначаться как «*q*»).

Разряд – это отдельная позиция в изображении числа в позиционной системе счисления.

Номер разряда – это номер позиции в записи числа, по сути, определяющий «вес» данного разряда. Разряды нумеруются, начиная с нуля.

Разрядность числа – это количество разрядов в записи числа, т. е. длина записи числа.

Нумерация разрядов (позиции символов) в числе идет от 0 до $N-1$ (где N – разрядность целой части) в целой части справа налево и от -1 до $-M$ (где M – разрядность дробной части) в дробной части записи числа слева направо. Пример обозначения номеров разрядов дробного десятичного числа представлен на рисунке 1.6.

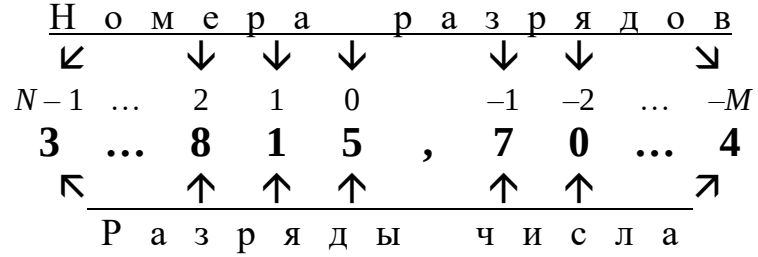


Рисунок 1.6 – Пример обозначения номеров разрядов дробного десятичного числа

Как в целой, так и в дробных частях самый левый разряд называется старшим, а самый правый – младшим.

Теоретически может существовать позиционная система счисления с любым основанием, большим чем нуль. Наиболее распространенные в настоящее время позиционные системы счисления представлены на рисунке 1.1.

В компьютерной технике используются двоичная, восьмеричная, десятичная и шестнадцатеричная системы счисления:

– в двоичной системе счисления происходит обработка информации в цифровых устройствах. Алфавит системы счисления: 0, 1. Основание системы счис-

ления $q_2 = 2$. Разряд в двоичной системе счисления принято называть битом. Восемь двоичных разрядов (8 бит) равны одному байту и представляют собой единицу измерения информации в компьютерной технике;

– восьмеричная система счисления в настоящее время по большей части вытеснена шестнадцатеричной, однако по-прежнему используется для определения прав доступа к файлам и прав исполнения в *Linux*-системах и в низкоуровневых языках программирования (языках ассемблера). Алфавит системы счисления: 0, 1, 2, 3, 4, 5, 6, 7. Основание системы счисления $q_8 = 8$;

– десятичная система счисления используется в операционных системах и языках программирования высокого уровня для ввода и вывода данных как привычная система счисления для человека. Алфавит системы счисления: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. Основание системы счисления $q_{10} = 10$;

– шестнадцатеричная система счисления в настоящее время используется в низкоуровневом программировании, при кодировке цветов в *RGB*-модели, обозначении ошибок выполнения операций и адресации пространства памяти. Алфавит системы счисления: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F. Основание системы счисления $q_{16} = 16$.

Двоичная, восьмеричная, десятичная и шестнадцатеричная системы счисления являются системами с равномерно распределенными ве́сами, которые характеризуются тем, что соотношение ве́сов двух любых соседних разрядов имеют для такой системы одинаковое значение. Это соотношение и определяется как *основание системы счисления*.

Числа в позиционной системе имеют сокращенную и расширенную формы записи.

Сокращенная запись числа N в системе с равномерно распределенными ве́сами имеет вид

$$N_q = A_n A_{n-1} \dots A_1 A_0. \tag{1.2}$$

Таким образом, примеры сокращенной записи чисел выглядят следующим образом:

– в десятичной системе счисления: 1989;

– в двоичной системе: 11111000101.

Количественная оценка числа, записанного в позиционной системе счисления, в общем виде определяется следующим образом:

$$\text{Количественная оценка} = \sum_{i=0}^{n} A_i \cdot q^i, \tag{1.3}$$

где n – количество разрядов в записи числа;

A_i – цифра записи числа ($0 \le A_i \le q - 1$);

q – основание системы счисления.

Расширенная запись числа выглядит следующим образом:

$$N_q = A_n \cdot q^n + A_{n-1} \cdot q^{n-1} + \ldots + A_1 \cdot q^1 + A_0 \cdot q^0. \qquad (1.4)$$

Пример расширенной записи числа 53,15 выглядит следующим образом:

– в десятичной СС: $53{,}15 = (5 \cdot 10^1 + 3 \cdot 10^0), (1 \cdot 10^{-1} + 5 \cdot 10^{-2})$;

– в двоичной СС: $110101{,}00100110 = (1 \cdot 2^5 + 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0), (0 \cdot 2^{-1} + 0 \cdot 2^{-2} + 1 \cdot 2^{-3} + 0 \cdot 2^{-4} + 0 \cdot 2^{-5} + 1 \cdot 2^{-6} + 1 \cdot 2^{-7} + 0 \cdot 2^{-8})$.

Исходя из вышеизложенного следует, что запись одного и того же числа в различных системах счисления будет тем длиннее (записана большим количеством разрядов), чем меньше основание системы счисления q:

$1110111110_{(\text{двоичная СС})} = 1676_{(\text{восьмеричная СС})} = 958_{(\text{десятичная СС})} = 3\text{BE}_{(\text{шестнадцатеричная СС})}$.

Поскольку все позиционные системы счисления имеют определенное пересечение алфавитов (используемых цифр), однозначно судить о том, в какой системе счисления записано число без специальных обозначений, не является однозначно правильным. Поэтому в разных средах применения существуют различные обозначения записей чисел в различных позиционных системах счисления. Примеры обозначений десятичного числа 12 (двенадцать) в разных системах счисления и в различных средах представлены в таблице 1.2.

Таблица 1.2 – Примеры обозначений записи числа 12 в различных СС и средах

Система счисления	В расчетах (на бумажном носителе)	В низкоуровневых языках программирования (ассемблер)			В языке программирования *C/C++*	В языке программирования *Delphi/Pascal*
Двоичная	1100_2	1100b			–	–
Восьмеричная	14_8	14o			x14	–
Десятичная	12_{10}	12			12	12
Шестнадцатеричная	C_{16}	Ch	0xC	$C	0xC	$C

Смешанные системы счисления являются разновидностью позиционных и представляют собой обобщение b-ичной системы счисления, основанием которой является возрастающая последовательность чисел $\{b_k\}_{k=0}^{\infty}$. Каждое число в такой системе счисления представляется линейной комбинацией

$$N = \sum_{k=0}^{n-1} A_k \cdot b_k, \qquad (1.5)$$

где A_k – цифры записи числа, на которые накладываются определенные ограничения, обусловленные конкретной смешанной системой счисления.

Смешанные системы счисления могут быть степенными, показательными и т. п., в зависимости от характера зависимости b_k от k.

Наиболее привычным примером смешанной системы счисления является исчисление времени в днях, часах, минутах и секундах. При этом величина «d дней, h часов, m минут, s секунд» соответствует значению $d \cdot 24 \cdot 60 \cdot 60 + h \times \times 60 \cdot 60 + m \cdot 60 + s$.

Факториальная и фибоначчиева системы счисления также являются разновидностью смешанных СС.

В факториальной системе счисления основанием является последовательность факториалов ($b_k = k!$), а числа представляются следующим образом:

$$N = \sum_{k=1}^{n} d_k \cdot k!, 0 \leq d_k \leq k. \qquad (1.6)$$

Фибоначчиева система счисления основывается на последовательности Фибоначчи и числа представляются следующим образом:

$$N = \sum_{k} f_k \cdot F_k, \qquad (1.7)$$

где $f_k \in \{0, 1\}$. В коэффициентах f_k есть конечное количество единиц, две единицы подряд не повторяются;

F_k – числа Фибоначчи.

Двоично-десятичная система счисления представляет собой синтез двоичной и десятичной систем счисления. Каждый разряд десятичного числа (каждая цифра) представляется в виде двоичной тетрады (четырех разрядов).

Пример записи числа $N = 3846$ в двоично-десятичной системе счисления:

N в десятичной СС:	3	8	4	6
N в двоично-десятичной СС:	0011	1000	0100	0110

$N_{10} = 3846 = N_{2-10} = 0011100001000110$.

Критерии выбора позиционных систем счисления

Существование множества позиционных систем счисления обусловлено их какими-либо преимуществами для применения в конкретных сферах.

Критериями выбора конкретной системы счисления для определенной сферы применения могут являться:

1 Наличие физических элементов для представления, хранения и обработки цифр системы счисления. Так, по данному критерию обусловлен выбор двоичной системы счисления в ЭВМ, поскольку полупроводниковые электрорадиоэлементы, магнитные элементы и т. д., на базе которых построены функцио-

нальные и операционные узлы ЭВМ, могут иметь только два устойчивых состояния (открыт/закрыт, заряжен/разряжен, намагничен / не намагничен и т. д.), которым можно поставить в соответствие значения нуль или единица.

2 Экономичность системы счисления определяет то количество чисел, которое можно записать в данной системе с помощью определенного количества цифр (общее количество сочетаний цифр, которые интерпретируются как различные числа), и обуславливает количество элементов, требуемых для представления и хранения многоразрядных чисел. Данный параметр может быть оценен показателем экономичности *C*.

Зависимость показателя экономичности *C* от основания системы счисления q имеет следующий вид:

$$C = q \cdot N, \tag{1.8}$$

где q – значение основания системы счисления;

N – длина разрядной сетки, выбранной для представления числа.

Максимальное число $A_{q_{\max}}$, которое можно представить в выбранной разрядной сетке, определяется как

$$A_{q_{\max}} = q \cdot N - 1. \tag{1.9}$$

Тогда длина разрядной сетки определяется как

$$N = \log_q\left(A_{q_{\max}} + 1\right). \tag{1.10}$$

Таким образом, показатель экономичности *C* любой системы счисления определяется как

$$C = q \cdot \log_q(A_{q_{\max}} + 1). \tag{1.11}$$

Приняв допущение, что значение q непрерывно (т. е. q может принимать вещественные значения), и приняв за единицу измерения оборудования условный элемент с одним устойчивым состоянием, для сравнения двух систем счисления можно ввести относительный показатель экономичности *F*.

При сравнении других систем счисления с используемой в компьютерной технике двоичной имеем

$$F = \frac{q \cdot \log_q(A_{q_{\max}} + 1)}{2\log_2(A_{2_{\max}} + 1)}. \tag{1.12}$$

Зависимость относительного показателя экономичности F от основания системы счисления при сравнении с двоичной системой счисления представлена на рисунке 1.7.

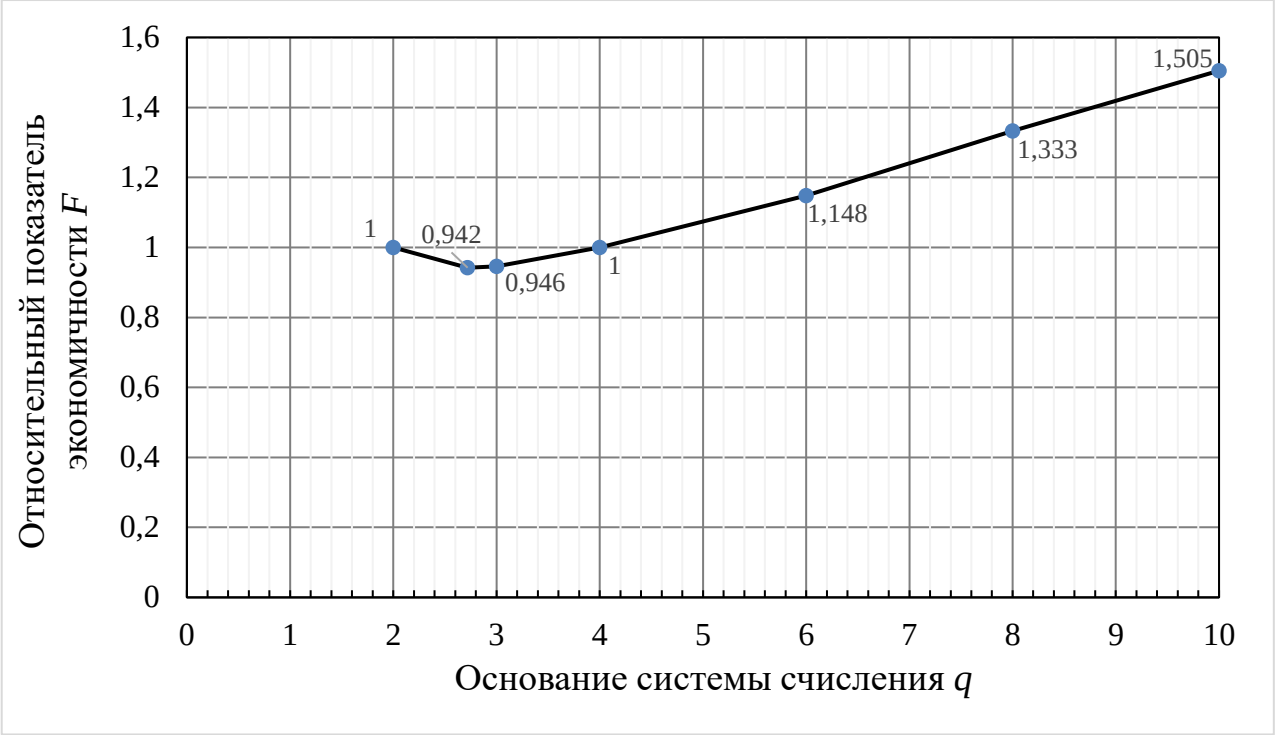


Рисунок 1.7 – Сравнение относительного показателя экономичности

Как видно из рисунка 1.7, с точки зрения затрат условного оборудования наиболее экономичной является система с основанием, равным $q \approx 2{,}718$ (локальный минимум). Поскольку основанием системы счисления может быть только целое число, наиболее близким к локальному минимуму является троичная система счисления, которая незначительно экономичнее двоичной и четверичной. Однако ее использование не получило распространения из-за отсутствия в электронике таких физических элементов, которые способны хранить три устойчивых состояния. Таким образом: «*Экономичность системы счисления ≠ Экономичность производства оборудования*».

Экономичность троичной системы счисления также можно наглядно проиллюстрировать, сравнив общее количество сочетаний цифр, которые можно интерпретировать как различные числа.

Например, возьмем 12 цифр (12 выбрано для примера с точки зрения большого количества делителей: 2, 3, 4, 6, 12, т. е. потенциальных систем счисления) и попробуем разбить их на разные группы: на шесть групп по две цифры (0, 1) для двоичной системы счисления (получив одно шестиразрядное двоичное число и максимум 2^6 уникальных комбинаций), на четыре группы по три цифры (0, 1, 2) для троичной (получив одно четырехразрядное троичное число и максимум 3^4 уникальных комбинаций), на три группы по четыре цифры (0, 1, 2, 3) для четверичной (получив одно трехразрядное четверичное число и максимум 4^3 уникальных комбинаций), на две группы по шесть цифр (0, 1, 2, 3, 4, 5) для шестеричной

(получив одно шестиразрядное шестеричное число и максимум 6^2 уникальных комбинаций) и на одну группу для двенадцатеричной системы счисления (одно двенадцатиразрядное число и максимум 12^1 уникальных комбинаций). Таким образом, число групп определит разрядность числа, а количество цифр в группе – основание системы счисления. Наглядное сравнение представлено в таблице 1.3 и рисунке 1.8.

Таблица 1.3 – Сравнение различных разбиений 12 исходных цифр

Основание системы счисления q	Разрядность числа N	Общее количество уникальных комбинаций (различных чисел) K
2	6	$2^6 = 64$
3	4	$3^4 = 81$
4	3	$4^3 = 64$
6	2	$6^2 = 36$
12	1	$12^1 = 12$

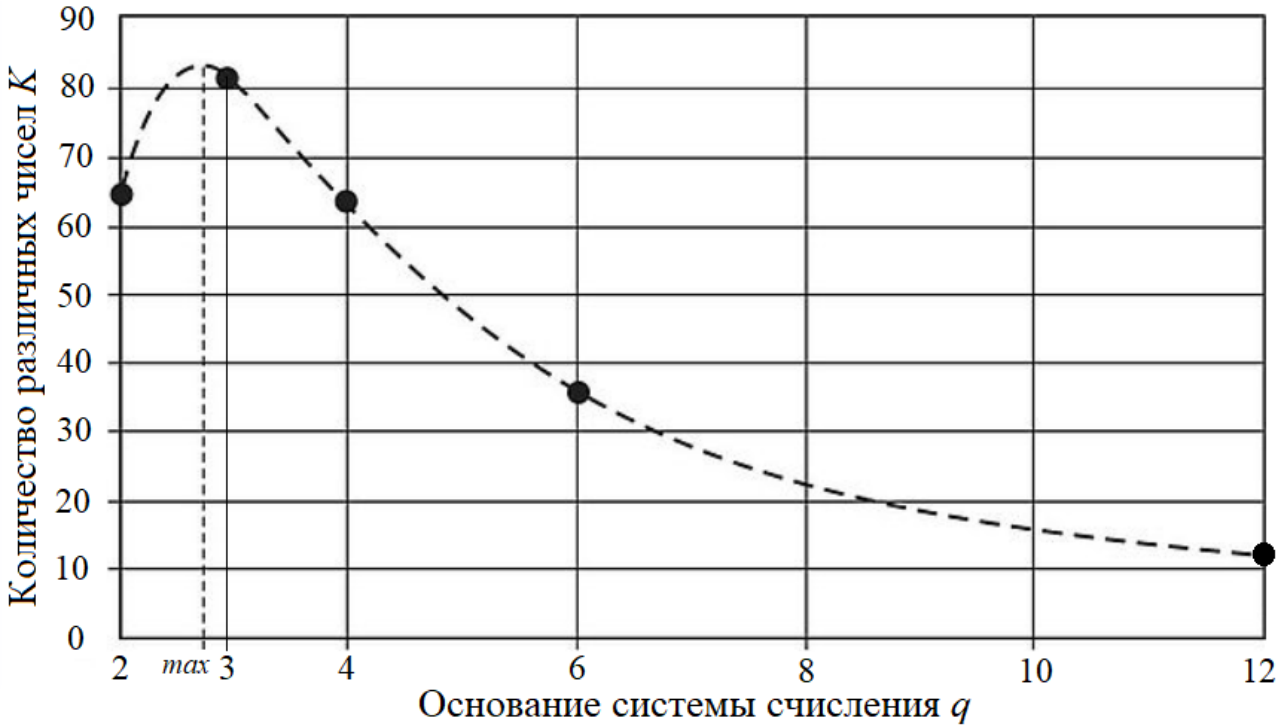


Рисунок 1.8 – Зависимость количества уникальных комбинаций от основания СС

Как видно из таблицы 1.3 и рисунка 1.8, троичная система счисления будет иметь наибольшую экономичность, т. к. при помощи нее можно представить большее количество различных чисел (уникальных комбинаций цифр системы счисления). Причем это будет справедливо и для случаев с другим исходным количеством цифр.

3 Помехоустойчивость определяет степень искажения действующего сигнала при наложении помех (шума). Очевидно, что при использовании систем счисления с меньшим основанием представление соседних цифр в этих системах отличается друг от друга больше, чем для систем с большим основанием, т. е. при увеличении основания помеха может привести к более существенному искажению действующего значения сигнала. Следовательно, системы с меньшим основанием имеют бóльшую помехоустойчивость по сравнению с системами с большим основанием.

4 Простота выполнения арифметических операций. Чем меньше основание системы счисления (меньше цифр в алфавите), тем проще правила выполнения арифметических действий над числами. Все возможные арифметические операции будут усложняться с увеличением основания системы счисления.

5 Быстродействие выполнения операций – многофакторный и довольно относительный критерий, зависящий в том числе и от основания системы счисления. Для упрощения рассмотрим его на примере выполнения арифметических операций. Как будет показано во втором разделе, все арифметические операции, выполняемые в компьютерной технике, можно так или иначе свести к операции алгебраического сложения, которая, в свою очередь, чаще всего сводится к арифметическому сложению. Таким образом, приняв арифметическое сложение за базовую операцию, можно определить абсолютное соотношение времени, необходимое для сложения, следующим образом:

$$T_{\text{СЛОЖ}} = t_{\text{пер}} \cdot N = t_{\text{пер}} \cdot \log_q N, \tag{1.13}$$

где N – разрядность числа;

q – основание системы счисления.

Тогда относительное значение оценки выполнения операции сложения

$$\sigma = \frac{T_{\text{СЛОЖ max}}}{T_{\text{СЛОЖ}}}. \tag{1.14}$$

При сравнении выполнения операции сложения в двоичной системе счисления с другими системами счисления имеем

$$T_{\text{СЛОЖ max}} = T_{\text{СЛОЖ (2)}} = t_{\text{пер}} \cdot \log_2 N. \tag{1.15}$$

Тогда σ определяется как

$$\sigma = \frac{t_{\text{пер}} \cdot \log_2 N}{t_{\text{пер}} \cdot \log_q N} = \log_2 q. \tag{1.16}$$

Таким образом, очевидно, что относительное значение оценки выполнения операции арифметического сложения (в сравнении с выполнением в двоичной системе счисления) будет возрастать с увеличением основания системы счисления (рисунок 1.9).

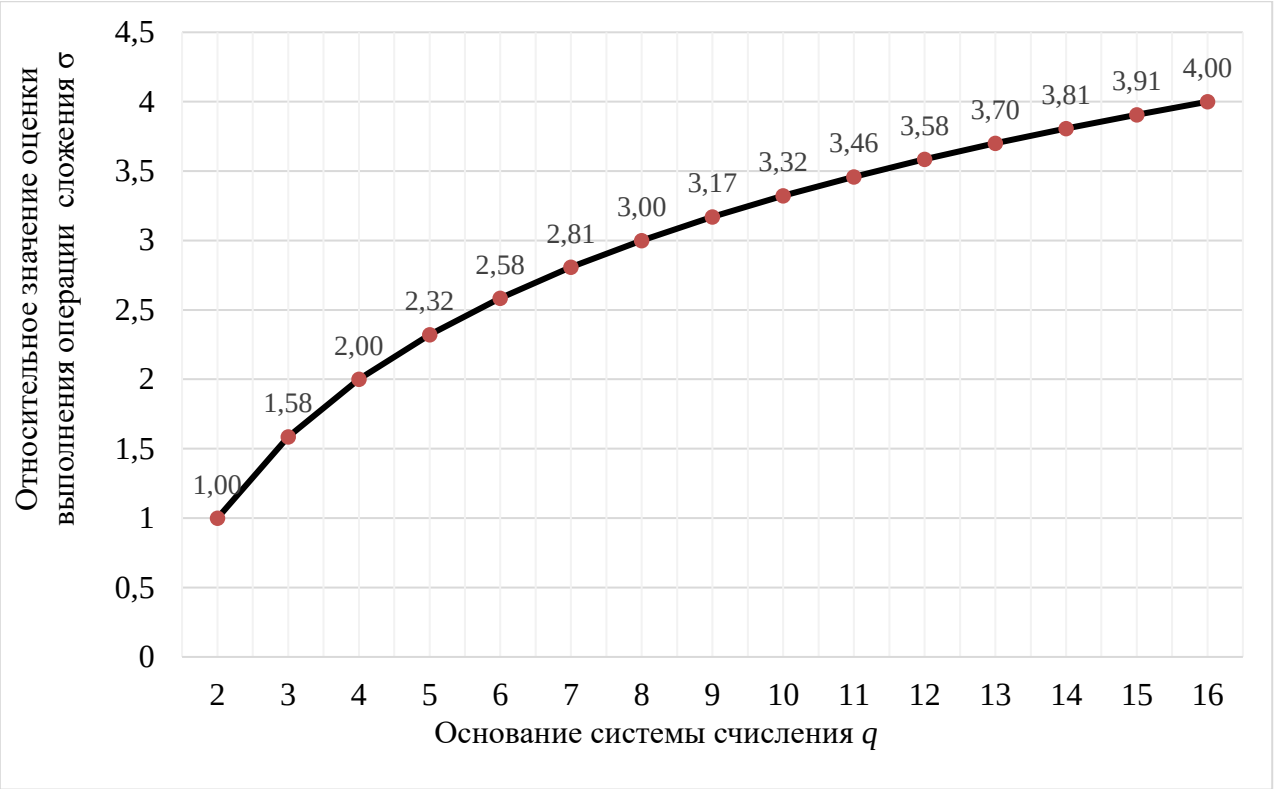


Рисунок 1.9 – Относительные значения оценки выполнения операции сложения

Таким образом, из указанных выше пояснений очевидно, что по совокупности всех критериев оптимальной системой счисления для представления, хранения и обработки информации в компьютерной технике является двоичная.

1.2 Методы перевода чисел из одной системы счисления в другую

Существование различных систем счисления предполагает необходимость перевода записи числа из одной системы счисления в другую. Существует ряд методов преобразований чисел в различных системах счисления:

– метод преобразования с использованием вéсов разрядов в заданной и в искомой записи числа;

– метод деления (умножения) на новое основание;

– метод преобразования по схеме Горнера;

– метод с использованием особого соотношения оснований заданной и искомой систем счисления.

Метод преобразования с использованием вéсов разрядов

Метод предполагает использование расширенной записи числа (1.4) в некоторой системе счисления и, в зависимости от того, какая система счисления (заданная или искомая) является более привычной, имеет две разновидности:

1 Если более привычной (десятичной) является искомая система счисления, то на основании расширенной записи заданного числа подсчитываются значения ее отдельных разрядов в искомой системе счисления. Далее полученные значения суммируются.

Пример

Необходимо перевести из двоичной системы счисления в десятичную число $N_2 = 1100110$. По формуле (1.4) формируем расширенную запись числа:

$$N_2 = 1100110 = 1 \cdot 2^6 + 1 \cdot 2^5 + 0 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0.$$

Далее рассчитывается вес двоичных разрядов в десятичной системе счисления:

$$N_2 = 1100110 = 64 + 32 + 0 + 0 + 4 + 2 + 0 = 102_{10}.$$

Искомое число в десятичной системе счисления $N_{10} = 102$.

При преобразовании правильных дробей используется тот же алгоритм, но при расчете вéсов отдельных разрядов берутся отрицательные степени основания счисления. Помимо этого, необходимо учитывать, что при преобразовании правильных дробей в общем случае результат получается неточный и перед началом преобразования необходимо подсчитать количество разрядов представления числа в новой системе счисления. Разрядность результата выбирается таким образом, чтобы ошибка представления результата была не более половины единицы младшего разряда в исходной записи числа. Таким образом, при переводе из двоичной в десятичную систему счисления берется соотношение, согласно которому *один десятичный разряд соответствует точности представления четырехразрядным двоичным числом (и наоборот)*.

При преобразовании правильных дробей сначала ищется предварительное значение представления заданного числа в новой системе счисления с количеством разрядов на единицу большим, чем расчетная разрядность представления числа в новой системе счисления. Дополнительный разряд в предварительном результате преобразования используется для округления, позволяющего с рассчитанным числом разрядов найти окончательный результат.

Пример

Необходимо перевести правильную двоичную дробь $N_2 = 0{,}111$ в правильную десятичную.

Перед началом преобразования определяется, что разрядность записи заданного числа в новой системе счисления должна быть равна единице, поэтому сначала ищется предварительная запись заданного числа в новой системе счисления с двумя или более двоичными разрядами:

$$N_2 = 0{,}111 = 1 \cdot 2^{-1} + 1 \cdot 2^{-2} + 1 \cdot 2^{-3} = 0{,}5 + 0{,}25 + 0{,}125 = 0{,}875.$$

После округления получаем $N_2 = 0{,}111 = 0{,}9_{10}$.

2 Если более привычной (десятичной) является заданная система счисления, то запись заданного числа в новой системе счисления определяется разряд за разрядом, начиная со старшего. Первым значащим разрядом будет являться разряд с максимальным возможным вéсом, но не превышающим значения преобразуемого числа. При этом, определив старший разряд с ненулевым значением, из исходного числа вычитаем вес этого разряда, таким образом формируя остаток, который должен быть представлен еще не найденным младшим разря-

дом искомой записи числа в новой системе счисления. Далее, используя полученный остаток, аналогичным способом ищется следующий разряд записи числа в новой системе счисления, определяется новый остаток и т. д.

Пример

Необходимо перевести из десятичной системы счисления в двоичную число $N_{10} = 234$.

Первоначально определяется вес старшего разряда таким образом, чтобы основание искомой системы счисления (2), возведенное в целую степень, равную весу старшего разряда, не превышало число в заданной системе счисления (234), но было максимально близко к нему. В данном случае $2^8 = 256$ превышает заданное число 234, а $2^7 = 128$ не превышает и максимально близко к нему.

Старший разряд с весом $2^7 = 128$ будет иметь значение «1» в искомой двоичной записи числа. С помощью последующих (младших) разрядов искомой записи числа необходимо представить значение 106, т. к. 106 – это остаток, полученный как разность между числами 234 и 128.

Следующий (слева направо) разряд с весом $2^6 = 64$ будет иметь в искомой двоичной записи числа значение «1». С помощью остальных (более младших) разрядов искомой записи числа необходимо представить значение 42, т. к. 42 – это остаток, полученный как разность между числами 106 и 64.

Следующий (слева направо) разряд с весом $2^5 = 32$ будет иметь в искомой двоичной записи числа значение «1», а остаток, определяемый как разность между числами 42 и 32, будет равен десяти.

Следующий (слева направо) разряд с весом $2^4 = 16$ будет иметь в искомой двоичной записи числа значение «0», а остаток остается прежним – равным десяти.

Следующий (слева направо) разряд с весом $2^3 = 8$ будет иметь в искомой двоичной записи числа значение «1», а остаток, определяемый как разность между числами 10 и 8, равен двум.

Следующий (слева направо) разряд с весом $2^2 = 4$ будет иметь в искомой двоичной записи числа значение «0», а остаток остается прежним – равным двум.

Следующий (слева направо) разряд с весом $2^1 = 2$ будет иметь в искомой двоичной записи числа значение «1», а остаток, определяемый как разность между числами 2 и 2, равен нулю.

Младший (последний) разряд с весом $2^0 = 1$ будет иметь в искомой двоичной записи числа значение «0», остаток равен нулю.

Таким образом, записывая полученные значения последовательно со старшего разряда, получаем

$234_{10} = 11101010_2$.

Коротко данное преобразование по шагам можно записать так:

Дано: $N_{10} = 234$.

Найти: $N_2 = ?$

1 Подбор веса старшего разряда (целой степени, при возведении в которую основания искомой системы счисления будет получено значение, не превышающее (меньшее или равное, но не большее) значения числа в заданной системе счисления).

1.1 Выбираем весом старшего разряда целое число **8**:
$2^8 = 256$, $256 > 234$ – превышает значение заданного числа => не подходит.

1.2 Выбираем весом старшего разряда целое число (меньше предыдущего) **7**:
$2^7 = 128$, $128 < 234$ – не превышает значение заданного числа и будет максимально близко к нему => определен вес старшего разряда искомой записи числа.

2 Определяем значения разрядов в искомой системе счисления (двоичной).

2.1 От значения числа в заданной системе счисления (234) отнимаем значение основания искомой системы счисления, возведенное в степень, равную подобранному весу разряда (2^7), и получаем остаток и первый единичный старший разряд записи искомого числа:

$234 - 2^7 = 234 - 128 = 106$	Старший разряд искомого числа = **1**

2.2 От полученного остатка отнимаем значение основания искомой системы счисления, возведенное в степень, равную весу подобранного разряда минус один (2^6). Далее на каждом шаге уменьшаем степень вплоть до нулевой (2^0) и каждый раз получаем новый остаток. Если остаток положительный или нуль, то записывается очередной единичный разряд записи искомого числа, если отрицательный, то очередной нулевой разряд:

$106 - 2^6 = 106 - 64 = 42$	Очередной разряд искомого числа = **1**
2.3 $42 - 2^5 = 42 - 32 = 10$	Очередной разряд искомого числа = **1**
2.4 $10 - 2^4 = 10 - 16$ (**отрицательный**)	Очередной разряд искомого числа = **0**

Поскольку получен отрицательный остаток, он не высчитывается, а на следующем шаге берется последний полученный положительный остаток.

2.5 $10 - 2^3 = 10 - 8 = 2$	Очередной разряд искомого числа = **1**
2.6 $2 - 2^2 = 2 - 4$ (**отрицательный**)	Очередной разряд искомого числа = **0**
2.7 $2 - 2^1 = 2 - 2 = 0$	Очередной разряд искомого числа = **1**

Несмотря на то что получен нулевой остаток, шаги продолжаются до достижения нулевого веса разряда.

2.8 $0 - 2^0 = 0 - 1$ (**отрицательный**)	Младший разряд искомого числа = **0**

2.9 Искомая запись числа формируется в порядке, прямом порядку получения разрядов (от старшего к младшему). Таким образом:
$N = 234_{10} = 11101010_{2.}$

Аналогично данный способ применяется при переводе правильных дробей, только с отрицательными степенями основания.

При переводе из десятичной в двоичную систему счисления берется соотношение, согласно которому *четыре двоичных разряда соответствуют точности представления одного десятичного разряда*.

Пример

Необходимо перевести правильную десятичную дробь $N_{10} = 0{,}7$ в правильную дробь в двоичной системе счисления.

Предварительный результат ищется с точностью до пяти двоичных разрядов, причем пятый разряд используется только для округления при переходе к четырехразрядному окончательному результату.

Старший разряд весом $2^{-1} = 0{,}5$ искомой двоичной записи числа будет иметь значение «1». С помощью остальных (младших) разрядов искомой записи числа необходимо представить значение 0,2 (0,2 – остаток, полученный как разность чисел 0,7 и 0,5).

Следующий (слева направо) разряд с весом $2^{-2} = 0{,}25$ в искомой двоичной записи числа будет иметь значение «0». Остаток остается прежним.

Следующий (слева направо) разряд с весом $2^{-3} = 0{,}125$ в искомой двоичной записи числа будет иметь значение «1». С помощью остальных (более младших) разрядов искомой записи числа необходимо представить значение 0,075 (0,075 – остаток, полученный как разность чисел 0,20 и 0,125).

Следующий (слева направо) разряд с весом $2^{-4} = 0{,}0625$ в искомой двоичной записи числа будет иметь значение «1», а остаток – 0,0125 (0,0125 – остаток, полученный как разность чисел 0,075 и 0,0625).

Младший (последний) разряд с весом $2^{-5} = 0{,}03125$ искомой двоичной записи числа будет иметь значение «0».

Таким образом, записывая полученные значения последовательно со старшего разряда, получаем $0{,}7_{10} = 0{,}10110_2$.

После округления (в соответствии с точностью) последнего младшего разряда имеем $0{,}7_{10} = 0{,}1011_2$.

Метод деления (умножения) на новое основание

Данный метод также предполагает использование расширенной записи числа (1.4) и имеет две разновидности: для целых и дробных чисел.

1 Преобразование целых чисел

Задачу представления числа N, заданного в системе q_1, в системе счисления с основанием q_2 можно рассматривать как задачу поисков коэффициентов полинома, представляющего собой расширенную запись числа N в системе счисления q_2:

$$N_{q1} = A_0 + A_1 \cdot q_2^1 + A_2 \cdot q_2^2 + \cdots + A_{n-1} \cdot q_2^{n-1} + A_n \cdot q_2^n = N_{q2}. \qquad (1.17)$$

Преобразуем выражение (1.17), вынеся общий знаменатель за скобку (рисунок 1.10).

$$N_{q1} = N_{q2} = A_0 + q_2\left(A_1 + q_2\left(A_3 + \ldots + q_2(A_{n-1} + q_2(A_n)\ldots)\right)\right)$$

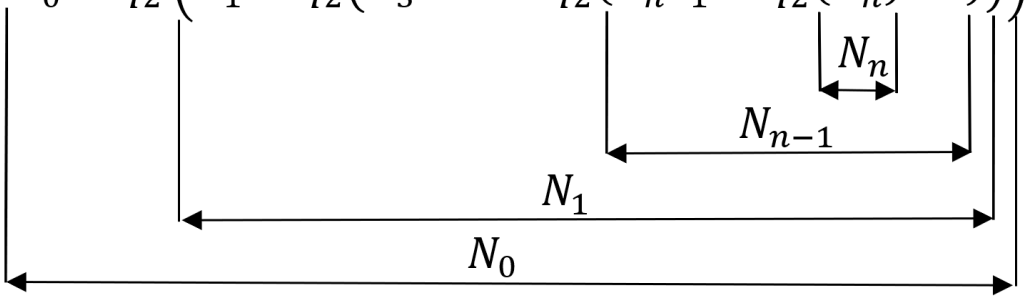


Рисунок 1.10 – Преобразованное выражение N_{q1}

Обозначим все преобразованное выражение как N_0, выражение в первой скобке как N_1, выражение во второй скобке как N_2 и т. д., выражение в $(n-1)$-й скобке – как N_{n-1}, выражение в n-й скобке – как N_n. Теперь, принимая во внимание выражении (1.8), можно утверждать, что при делении N_{q1} на q_2 будет получена целая часть частного (обозначим ее int (N_{q1} / q_2)) и остаток (обозначим его rest (N_{q1} / q_2)), равный A_0. Тогда остальные скобки будут иметь вид:

$\frac{N_1}{q_2}$: целая часть int (N_1 / q_2), равная N_2, и остаток rest (N_1 / q_2), равный A_1;

$\frac{N_2}{q_2}$: целая часть int (N_2 / q_2), равная N_3, и остаток rest (N_2 / q_2), равный A_2;

…

$\frac{N_{n-2}}{q_2}$: целая часть int (N_{n-2} / q_2), равная N_{n-1}, и остаток rest (N_{n-2} / q_2), равный A_{n-2};

$\frac{N_{n-1}}{q_2}$: целая часть int (N_{n-1} / q_2), равная $N_n = A_n$, и остаток rest (N_{n-1} / q_2), равный A_{n-1}.

При этом $N_n = A_n$ $(N_n < q_2)$.

Отсюда следует правило формирования коэффициентов полинома (1.17), которые и являются разрядами записи заданного числа N в системе счисления с основанием q_2:

– необходимо разделить исходное число N_{q1} на новое основание q_2, при этом получив целое частное и остаток;

– полученный остаток снова необходимо разделить на q_2, процесс деления продолжается до тех пор, пока частное будет не меньше нового основания q_2;

– если очередное сформированное частное будет меньше, чем q_2, то процесс формирования записи заданного числа в новой системе с основанием q_2 считается законченным, а в качестве искомых разрядов новой записи числа используются результаты выполненных операций деления следующим образом: *в качестве старшего разряда берется значение последнего частного, для остальных разрядов используются значения остатков в порядке, обратном порядку их получения.*

Пример

Перевести десятичное число $N_{10} = 432$ в двоичную систему счисления методом деления на новое основание.

Согласно алгоритму, сначала делим исходное число N_{10}, а затем и получаемые частные на значение нового основания $q = 2$ до получения частного со значением, меньшим чем новое основание (2):

$\frac{432}{2}$: int (432 / 2) = 216 и rest (432 / 2) = **0**;

$\frac{216}{2}$: int (216 / 2) = 108 и rest (216 / 2) = **0**;

$\frac{108}{2}$: int (108 / 2) = 54 и rest (108 / 2) = **0**;

$\frac{54}{2}$: int (54 / 2) = 27 и rest (54 / 2) = **0**;

$\frac{27}{2}$: int (27 / 2) = 13 и rest (27 / 2) = **1**;

$\frac{13}{2}$: int (13 / 2) = 6 и rest (13 / 2) = **1**;

$\frac{6}{2}$: int (6 / 2) = 3 и rest (6 / 2) = **0**;

$\frac{3}{2}$: int (3 / 2) = **1** и rest (3 / 2) = **1**.

Далее, как показано стрелками, формируем искомую запись числа в двоичной системе счисления. Таким образом,

$N_{10} = 432 = 10110000_2$.

2 Преобразование дробных чисел

Задачу представления дробного числа M_{q1} в новой системе счисления с основанием q_2 можно рассматривать как задачу поиска коэффициентов полинома, представляющего собой расширенную запись числа M в системе счисления q_2:

$$M_{q1} = B_1 \cdot q_2^{-1} + B_2 \cdot q_2^{-2} + \ldots + B_{n-1} \cdot q_2^{-(n-1)} + B_n \cdot q_2^{-n} = M_{q2}. \qquad (1.18)$$

Преобразуем выражение (1.18), вынеся общий знаменатель за скобку (рисунок 1.11).

$$M_{q1} = M_{q2} = q_2^{-1}\left(B_1 + q_2^{-1}\left(B_2 + q_2^{-1}(B_3 + \ldots + q_2^{-1}(B_{n-1} + q_2^{-1}(B_n))\ldots)\right)\right)$$

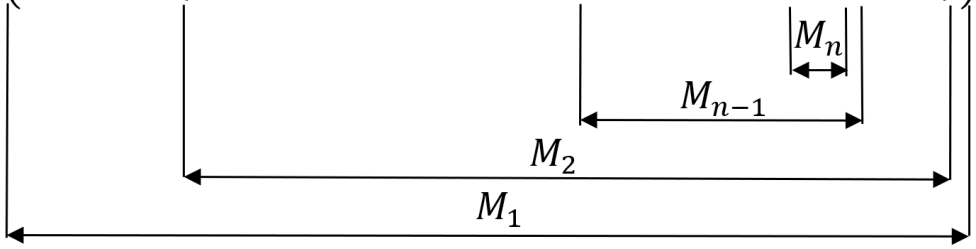


Рисунок 1.11 – Преобразованное выражение M_{q1}

Обозначим выражение в первой скобке как M_1, выражение во второй скобке – как M_2 и т. д., выражение в $(n-1)$-й скобке – как M_{n-1}, выражение в n-й скобке – как M_n.

Число M_{q1} – правильная дробь, при умножении M_{q1} на основание q_2 будет получено произведение, в общем случае состоящее из целой части int $(M_{q1} \cdot q_2)$ и дробной части DF $(M_{q1} \cdot q_2)$. Тогда остальные выражения в скобках будут иметь вид:

$M_1 \cdot q_2 = (\text{int}\ (M_1 \cdot q_2) = B_2) + (\text{DF}\ (M_1 \cdot q_2) = M_2)$;

$M_2 \cdot q_2 = (\text{int}\ (M_2 \cdot q_2) = B_3) + (\text{DF}\ (M_2 \cdot q_2) = M_3)$;

…

$M_{n-1} \cdot q_2 = (\text{int}\ (M_{n-1} \cdot q_2) = B_n) + (\text{DF}\ (M_{n-1} \cdot q_2) = M_n)$;

$M_n \cdot q_2 = (\text{int}\ (M_n \cdot q_2) = B_{n+1}) + (\text{DF}\ (M_n \cdot q_2) = M_{n+1})$.

Отсюда следует правило формирования коэффициентов полинома, которые и являются разрядами записи заданного числа M в системе счисления с основанием q_2:

– определяется количество разрядов n в записи числа M_{q2} в новой системе счисления. Разрядность результата выбирается таким образом, чтобы ошибка представления результата была не более половины единицы младшего разряда в исходной записи числа;

– исходное число M_{q1} умножается на q_2, при этом будет получено смешанное число;

– дробная часть полученного произведения снова умножается на q_2 и т. д. Процесс умножения повторяется n раз;

– в качестве искомых разрядов новой записи числа используются результаты выполненных операций умножения следующим образом: в качестве первого старшего разряда искомой записи числа в системе счисления с новым основанием берется значение целой части первого произведения, в качестве следующего (слева направо) разряда искомой записи числа в системе счисления с новым основанием берется значение целой части второго произведения и т. д.

Пример

Перевести десятичное число $M_{10} = 0{,}7$ в двоичную систему счисления методом умножения на новое основание.

Сначала необходимо определить количество разрядов дробной части числа M в двоичной системе счисления (целая часть равна нулю, т. к. это правильная дробь). Так как исходная запись числа содержит один десятичный разряд, то запись данного числа в двоичном основании должна содержать четыре разряда. Учитывая округление, ищется предварительный двоичный эквивалент с пятью разрядами.

Умножаем исходное число M_{10}, а затем дробные части последовательно получаемых произведений на новое основание $q = 2$:

0,7 · 2 = 1,4 (int (0,7 · 2) = 1 и DF (0,7 · 2) = 0,4);
0,4 · 2 = 0,8 (int (0,4 · 2) = 0 и DF (0,4 · 2) = 0,8);
0,8 · 2 = 1,6 (int (0,8 · 2) = 1 и DF (0,8 · 2) = 0,6);
0,6 · 2 = 1,2 (int (0,6 · 2) = 1 и DF (0,6 · 2) = 0,2);
0,2 · 2 = 0,4 (int (0,2 · 2) = 0 и DF (0,2 · 2) = 0,4).

Далее, как показано стрелкой, формируем искомую запись числа в двоичной системе счисления.

Таким образом,

$M_{10} = 0,7 = 0,10110_2$.

После округления (в соответствии с точностью) имеем

$M_{10} = 0,7_{10} = 0,1011_2$.

Перевод из одной системы счисления в другую неправильных дробей

Для перевода дробных чисел, имеющих ненулевую целую часть (неправильная дробь), необходимо по правилам, описанным выше, отдельно переводить целую часть и отдельно – дробную.

Метод преобразования по схеме Горнера

Схема Горнера в общем случае представляет собой способ деления многочлена на бином.

Применительно к переводу чисел из одной системы счисления в другую схема Горнера позволяет минимизировать арифметические операции и исключить возведение в степень. Она применима при переводе чисел в десятичную систему счисления. Для этого необходимо преобразовать расширенную запись числа по схеме Горнера, в результате чего перевод числа сводится к выполнению последовательности операций умножения и сложения в порядке записи числа слева направо.

Для целого n-разрядного числа в p-ичной системе счисления преобразование по схеме Горнера имеет следующий вид:

$$N_p = A_{n-1}A_{n-2}\dots A_0 = A_{n-1}\cdot p^{n-1} + A_{n-2}\cdot p^{n-2} + \dots + A_0\cdot p^0 = p(\dots p(pA_{n-1} + A_{n-2}) + \dots + A_1) + A_0. \quad (1.19)$$

Полученное в ходе преобразования равенство будет справедливо для любых целых p-ичных чисел. Формула преобразования в общем виде

$$A_nA_{n-1}A_{n-2}\dots A_0 = (((\dots(A_n\cdot p + A_{n-1})p + A_{n-2})p + \dots) + A_1)p + A_0. \quad (1.20)$$

Алгоритм перевода чисел по схеме Горнера можно сформулировать так:

– цифру старшего разряда умножаем на основание;

– добавляем цифру следующего за старшим разряда;

– результат умножаем на основание;

– добавляем цифру следующего разряда и так до тех пор, пока не прибавим цифру последнего младшего разряда. Результатом будет искомая десятичная запись числа.

Для дробного (правильная дробь) n-разрядного числа в p-ичной системе счисления преобразование по схеме Горнера имеет следующий вид:

$$N_p = 0, A_1 A_2 \dots A_n = A_1 \cdot p^{-1} + A_2 \cdot p^{-2} + \dots + A_n \cdot p^{-n} = \\ = \left(\dots \left(\frac{A_n}{p} + A_{n-1} \right) \frac{1}{p} + \dots + A_1 \right) \frac{1}{p}. \quad (1.21)$$

Пример

Перевести троичное число 20121_3 в десятичную систему счисления.

Формируется расширенная запись троичного числа:

$20121_3 = 2 \cdot 3^4 + 0 \cdot 3^3 + 1 \cdot 3^2 + 2 \cdot 3^1 + 1 \cdot 3^0$.

Выполняется преобразование расширенной записи по схеме Горнера.

Поскольку $3^0 = 1$ (любое натуральное число, возведенное в нулевую степень равно единице), то только первые четыре слагаемых имеют общий множитель, равный трем, который можно вынести за скобки.

Тогда:

$2 \cdot 3^4 + 0 \cdot 3^3 + 1 \cdot 3^2 + 2 \cdot 3^1 + 1 \cdot 3^0 = (2 \cdot 3^3 + 0 \cdot 3^2 + 1 \cdot 3^1 + 2) \cdot 3 + 1$.

Теперь первые три слагаемых также имеют общий множитель, равный трем, который аналогично выносим за скобки:

$(2 \cdot 3^3 + 0 \cdot 3^2 + 1 \cdot 3^1 + 2) \cdot 3 + 1 = ((2 \cdot 3^2 + 0 \cdot 3^1 + 1) \cdot 3 + 2) \cdot 3 + 1$.

Теперь первые два слагаемых имеют общий множитель – три, который аналогично выносим за скобки:

$((2 \cdot 3^2 + 0 \cdot 3^1 + 1) \cdot 3 + 2) \cdot 3 + 1 = (((2 \cdot 3 + 0) \cdot 3 + 1) \cdot 3 + 2) \cdot 3 + 1$.

Вычисляем выражение:

$(((2 \cdot 3 + 0) \cdot 3 + 1) \cdot 3 + 2) \cdot 3 + 1 = 178$.

Таким образом,

$20121_3 = 178_{10}$.

Метод преобразования с использованием особого соотношения оснований заданной и искомой систем счисления

Метод не является универсальным и применим тогда, когда исходное q_1 и новое q_2 основания могут быть связаны через целую степень, т. е. когда выполняется одно из двух условий формулы

$$q_2^m = q_1; \\ q_1^m = q_2. \quad (1.22)$$

Условие q_2^m (1.22) применимо для перехода из системы с большим основанием в систему с меньшим основанием. Условие q_1^m (1.22) применимо для перехода из системы с меньшим основанием в систему с бóльшим основанием.

Если выполняется условие q_2^m, тогда запись числа $N_{q1} = a_n a_{n-1} \dots a_1 a_0$ в системе с новым основанием q_2 определяется следующим образом:

– каждому разряду a_i исходной записи числа ставится в соответствие его m-разрядный эквивалент в системе счисления с основанием q_2;

– исходная запись всего заданного числа формируется за счет объединения всех полученных m-разрядных групп.

Пример

Перевести дробное восьмеричное число $N_8 = 47601{,}62$ в двоичную систему счисления.

Основания исходной и новой систем счисления можно выразить через целую степень следующим образом: $2^3 = 8$. Согласно алгоритму ставим в соответствие каждой цифре исходной записи восьмеричного числа трехразрядный двоичный эквивалент – *триады* (таблица 1.4).

Таблица 1.4 – Соответствие двоичных триад цифрам восьмеричного числа

Описание	Цифровое выражение разрядов и триад							
Представление цифр восьмеричного числа	4	7	6	0	1	,	6	2
Соответствие триад в двоичной системе счисления	100	111	110	000	001	,	110	010

Формируем окончательный результат посредством объединения полученных триад двоичного представления восьмеричных цифр в единый двоичный эквивалент с учетом разделителя, отделяющего целую часть числа от дробной:

$47601{,}62_8 = 100111110000001{,}110010_2$.

Если выполняется условие q_1^m, то запись числа $N_{q1} = a_n a_{n-1} \dots a_1 a_0$ в системе с новым основанием q_2 определяется следующим образом:

– исходная запись числа разбивается на группы по m разрядов, двигаясь от разделителя целой и дробной части (запятой) вправо и влево (недостающие разряды в крайних группах (слева и справа) дополняются нулями);

– каждой полученной m-разрядной группе ставится в соответствие цифра новой системы счисления;

– искомая запись заданного числа в новой системе счисления образуется из цифр, соответствующих группам, на которые была разбита исходная запись.

Пример

Перевести дробное двоичное число $N_2 = 1101111100{,}1110100$ в шестнадцатеричную систему счисления.

Основания исходной и новой систем счисления можно выразить через целую степень следующим образом: $2^4 = 16$. Разбиваем исходную запись числа на группы по четыре разряда (*тетрады*) вправо и влево от запятой, в крайних левой и правой группах недостающие разряды заполняем нулями и каждой полученной группе из четырех разрядов ставим в соответствие цифру шестнадцатеричной системы счисления (таблица 1.5).

Таблица 1.5 – Соответствие двоичных тетрад цифрам шестнадцатеричного числа

Описание	Цифровое выражение тетрад и разрядов					
Представление тетрад в двоичной системе счисления	**00**11	0111	1100	,	1110	100**0**
Соответствие цифр шестнадцатеричного числа	3	7	C	,	E	8

В таблице 1.5 полужирным подчеркнутым шрифтом обозначены недостающие разряды в крайней правой и крайней левой от запятой тетрадах (группа из четырех разрядов), которые заполняются нулями.

Формируем окончательный результат посредством объединения полученных цифр в единый шестнадцатеричный эквивалент с учетом запятой, отделяющей целую часть от дробной:

$1101111100{,}1110100_2 = 37C{,}E8_{16}$.

Следует также отметить, что применять метод преобразования с использованием особого соотношения оснований иногда бывает целесообразно, даже если основания исходной q_1 и новой q_2 систем счисления не могут быть связаны через целую степень напрямую. Если существует промежуточная система, которая может быть связана с заданной и искомой системами двумя условиями (1.22) одновременно, то можно сперва найти эквивалент заданного числа в промежуточной системе счисления, а затем из промежуточной системы перевести в искомую. В ряде случаев такие преобразования выполняются быстрее, нежели рассмотренными выше универсальными методами.

Пример

Перевести восьмеричное число $N_8 = 27514{,}73_8$ в шестнадцатеричную систему счисления.

Промежуточной системой счисления в данном случае может являться двоичная, т. к. согласно условиям (1.22) $2^3 = 8$ и $2^4 = 16$.

Сначала необходимо перевести число из заданной восьмеричной системы счисления в промежуточную двоичную (таблица 1.6).

Таблица 1.6 – Соответствие двоичных триад цифрам восьмеричного числа

Описание	Цифровое выражение разрядов и триад							
Представление цифр восьмеричного числа	2	7	5	1	4	,	7	3
Соответствие триад в двоичной системе счисления	010	111	101	001	100	,	111	011

Таким образом, имеем

$27514{,}73_8 = 010111101001100{,}111011_2$.

Далее переводим число из промежуточной двоичной системы счисления в искомую шестнадцатеричную, разбив в обе стороны от запятой запись двоичного числа на тетрады и дополнив крайнюю левую и крайнюю правую тетрады недостающими нулевыми разрядами при необходимости (полужирный подчеркнутый шрифт) (таблица 1.7).

Таблица 1.7 – Соответствие двоичных тетрад цифрам шестнадцатеричного числа

Описание	Цифровое выражение тетрад и разрядов						
Представление тетрад в двоичной системе	**0**010	1111	0100	1100	,	1110	11**00**
Соответствие цифр шестнадцатеричного числа	2	F	4	C	,	E	C

В итоге

$27514{,}73_8 = 010111101001100{,}111011_2 = 2F4C.EC_{16}$.

Следует отметить, что для целой части чисел незначащими (т. е. их можно отбросить и это не повлияет на количественную оценку числа) являются нулевые старшие разряды, а для дробной – незначащими являются нулевые младшие разряды в любых позиционных системах счисления.

Пример таких чисел представлен в таблице 1.8.

Таблица 1.8 – Числа, содержащие в записи незначащие разряды (перечеркнуты)

Число с нулевыми незначащими разрядами		Количественная оценка	
в десятичной СС	в двоичной СС	в десятичной СС	в двоичной СС
~~**000**~~7,05~~**00**~~	~~**000**~~111,000011001~~**00**~~	7,05	111,0000110011
12,125~~**000**~~	1100,001~~**000**~~	12,125	1100,001
~~**000000**~~58	~~**000000**~~111010	58	111010

1.3 Представление целых чисел в памяти компьютера

В компьютерной технике для представления и обработки целых алгебраических чисел используются код со сдвигом и следующие базовые коды:

– прямой;

– дополнительный;

– обратный.

Код со сдвигом (двоичное смещение / *offset binary*)

Целочисленный отрезок от 0 до 2n сдвигается влево на 2n − 1, затем получившиеся на этом отрезке числа последовательно кодируются в порядке возрастания кодами от 000…0 до 111…1.

Двоичное смещение чаще всего используется в цифровой обработке сигналов (*Digital Signal Processing*).

При арифметических операциях, выполняемых в коде со сдвигом, нужно учитывать смещение, т. е. проделывать на одно действие больше (например, после стандартной операции сложения двух чисел у результата будет двойное смещение, одно из которых необходимо вычесть).

Из-за необходимости усложнять арифметические операции код со сдвигом для представления целых чисел используется не часто, но зато применяется для хранения порядка вещественного числа.

Достоинства кода со сдвигом:

– не требуется усложнение архитектуры процессора;

– не возникает проблема двух нулей.

Недостатки кода со сдвигом:

– при арифметических операциях нужно учитывать смещение;

– ряд положительных и отрицательных чисел несимметричен.

Все три разновидности базового кодирования чисел (прямой, обратный и дополнительный) имеют формат представления, содержащий два поля: поле знака и поле модуля (рисунок 1.12).

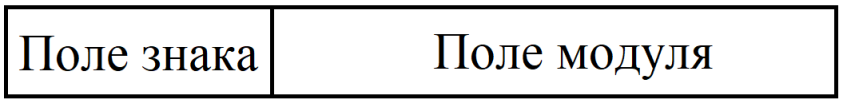


Рисунок 1.12 – Формат представление алгебраических двоичных чисел

Во всех разновидностях *базовой кодировки* поле знака представлено одним разрядом, в котором устанавливается нуль, если число положительное, и единица, если число отрицательное. Поле модуля для каждого кода формируется поразному. Количество разрядов поля модуля определятся диапазоном изменения отображаемых чисел (для целых чисел) или точностью их представления (для вещественных чисел).

Прямой код предназначен для хранения в памяти и представления чисел, поскольку только в нем модульное поле *содержит количественную оценку числа.*

Обратный и дополнительный коды позволяют упростить арифметические операции над алгебраическими числами. Поэтому обратный и дополнительный коды предназначены *для выполнения арифметических операций* над алгебраическими числами. В общем случае модульное поле в обратном/дополнительном коде *не отражает количественной оценки числа*.

Прямой базовый код

При кодировании числа в прямой код запись целого числа A формируется по следующему правилу:

$$[A]_{\text{ПК}} = \begin{cases} 0.A, \text{если } A \geq 0; \\ 1.|A|, \text{если } A < 0. \end{cases} \quad (1.23)$$

Обратный базовый код (дополнение до единицы / *one's complement*)

При кодировании числа в обратный базовый код запись целого числа A формируется по следующему правилу:

$$[A]_{\text{ОК}} = \begin{cases} 0.A, \text{если } A \geq 0; \\ 1.((q^n - 1) + A), \text{если } A < 0, \end{cases} \quad (1.24)$$

где q – основание системы счисления;

n – разрядность модульного поля;

$q^n - 1$ – максимальная включенная граница диапазона изменения представляемых чисел.

Достоинства обратного базового кода:

– коды положительных чисел относительно беззнакового кодирования остаются неизменными;

– количество положительных чисел равно количеству отрицательных.

Недостатки обратного базового кода:

– арифметические операции с отрицательными числами требуют усложнения архитектуры процессора;

– существует проблема двух нулей: нулю соответствуют коды 00...000 и 11...111.

Дополнительный базовый код (дополнение до двух / *two's complement*)

При кодировании числа в дополнительный базовый код запись целого числа A формируется по следующему правилу:

$$[A]_{\text{ДК}} = \begin{cases} 0.A, \text{если } A \geq 0; \\ 1.(q^n + A), \text{если } A < 0, \end{cases} \quad (1.25)$$

где q – основание системы счисления;

n – разрядность модульного поля;

q^n – максимальная невключенная граница диапазона изменения представляемых чисел.

Чаще всего дополнительный базовый код используется для представления отрицательных чисел.

По сути, дополнительный код представляет собой дополнение $|A|$ до нуля.

Достоинства дополнительного базового кода:

– возможность заменить арифметическую операцию вычитания операцией сложения и сделать операции сложения одинаковыми для знаковых и беззнаковых типов данных, что существенно упрощает архитектуру процессора и увеличивает быстродействие (см. раздел 2);

– нет проблемы двух нулей.

Недостатки дополнительного базового кода:

– ряд положительных и отрицательных чисел несимметричен;

– числа в дополнительном базовом коде нельзя сравнивать как беззнаковые или вычитать без расширения разрядности.

Несмотря на недостатки, в современных вычислительных системах при выполнении арифметических операций чаще всего используется дополнительный код.

Таким образом:

– положительное алгебраическое число в прямом, обратном и дополнительном кодах имеет одинаковое представление (совпадает);

– для того чтобы перевести отрицательное число из обратного в прямой код (и наоборот), необходимо дополнить его модуль до включенной границы;

– для того чтобы перевести отрицательное число из дополнительного в прямой код (и наоборот), необходимо дополнить его модуль до невключенной границы;

– для двоичных чисел включенной границей является значение, равное 2^n, а невключенной границей является значение, равное $2^n - 1$.

Следует обратить внимание, что обратный и дополнительный коды находятся только из прямого кода соответствующего операнда!

Пример нахождения обратного и дополнительного кода

Сформировать запись десятичных чисел $A = 182_{10}$ и $B = -107_{10}$ в прямом, обратном и дополнительном двоичных кодах

Сначала переводим десятичные числа A и B в двоичные:

$A = 182_{10} = +10110110_2$, $B = -107_{10} = -1101011_2$.

Далее формируем запись чисел в прямом, обратном и дополнительном кодах согласно (1.23), (1.24) и (1.25). Поскольку число A положительное, его представление модульного поля в обратном и дополнительном кодах будет совпадать с представлением в прямом коде, а знаковое поле обозначается положительным знаком (нуль):

$[A]_{ПК} = [A]_{ОК} = [A]_{ДК} = 0.10110110$.

Число B отрицательное и будет иметь следующее представление в прямом коде:

$[B]_{ПК} = 1.1101011$.

Для определения модульной части числа B в обратном коде прибавим к включенной границе диапазона ($2^n - 1 = 1111111$) число B по формуле (1.24):

$1111111 + (-1101011) = 0010100$.

Таким образом,

$[B]_{ОК} = 1.0010100$.

Для определения модульной части числа B в дополнительном коде прибавим к невключенной границе диапазона ($2^n = 10000000$) число B по формуле (1.25):

$10000000 + (-1101011) = 0010101$.

Таким образом,

$[B]_{ДК} = 1.0010101$.

Как видно из примера, запись модульного поля отрицательного числа в обратном коде представляет собой инверсию модульной части записи этого числа в прямом коде, т. е. нули заменяются единицами, а единицы заменяются нулями. В свою очередь, запись модульного поля отрицательного числа в дополнительном коде отличаются от записи в обратном коде на значение, соответствующее единице младшего разряда.

Исходя из (1.23–1.25) в общем виде *правила формирования модуля отрицательного целого двоичного числа* имеют вид:

– чтобы сформировать модульную часть записи отрицательного числа в обратном коде, необходимо в модульной части записи этого числа в прямом коде

взять обратные значения всех двоичных разрядов (т. е. *проинвертировать* модуль прямого кода: нули заменяются единицами, а единицы – нулями);

– для перевода отрицательного числа из обратного кода назад в прямой также необходимо *проинвертировать* запись модульной части числа в обратном коде;

– чтобы сформировать модульную часть записи отрицательного числа в дополнительном коде, необходимо в модульной части записи числа в прямом коде взять обратные значения всех двоичных разрядов (т. е. *проинвертировать* модуль прямого кода), и к полученному *прибавить единицу в младший разряд*;

– для перевода отрицательного числа из дополнительного кода назад в прямой также необходимо *проинвертировать* запись модульной части числа в дополнительном коде и к полученному коду *прибавить единицу в младший разряд*.

Модифицированные коды

Базовые коды не очень удобны при выполнении арифметических операций, поскольку при выполнении операции может произойти переполнение модульного поля, которое никак не определяется в базовом коде. Может сложиться абсурдная ситуация, когда при сложении двух отрицательных целых чисел получится положительная сумма.

Для исключения подобных ситуаций существуют модификации прямого, обратного и дополнительного кода.

Поле знака в модифицированных кодах представлено двумя разрядами по следующему правилу:

– 00 – для положительных чисел;

– 11 – для отрицательных чисел.

Если в результате выполнения арифметической операции разряды в знаковом поле останутся одинаковыми (00 или 11), то переполнение модульного поля не произошло.

Если в результате выполнения арифметической операции разряды в знаковом поле окажутся разными (01 или 10) – это будет свидетельствовать о переполнении модульного поля, причем:

– старший разряд является знакоопределяющим (0 – положительный, 1 – отрицательный);

– младший разряд несет в себе информацию не о знаке, а количественную оценку числа и, соответственно, должен быть перемещен в модульное поле.

Таким образом, при переполнении модульного поля возможны следующие варианты:

– 01 – переполнение поля модуля, число считается положительным, единичный разряд несет количественную оценку числа и должен быть перемещен в старший разряд модульного поля;

– 10 – переполнение поля модуля, число является отрицательным, нулевой разряд несет количественную оценку числа и должен быть перемещен в старший разряд модульного поля.

1.4 Представление вещественных чисел в памяти компьютера

В компьютерной технике вещественные алгебраические числа могут быть представлены в формате с фиксированной точкой (запятой) или в формате с плавающей точкой (запятой).

Вещественные числа в формате с фиксированной точкой также представляются в кодированной записи (прямой, обратный, дополнительный коды) и содержат поле знака и поле модуля. Поле модуля в данном случае хранит целую и дробные части, разделенную не обозначаемым, но подразумеваемым разделителем (точкой или запятой).

При таком формате представления вещественного числа положение разделителя строго фиксировано, т. к. порядок p числа постоянен. Выбор величины порядка p числа в общем случае произволен. При этом положение разделителя целой и дробной части закрепляется (только подразумевается и никак не обозначается) в определенном месте относительно всех разрядов числа и сохраняется неизменным для всех чисел, представляемых в выбранной разрядной сетке вычислительной машины.

Наиболее распространены два способа закрепления разделителя:

– перед старшим разрядом ($p = 0$); тогда все числа по модулю меньше единицы;

– после младшего разряда ($p = n$), где n – количество разрядов цифровой части числа; тогда представляемые числа знаковые целые (с нулевой дробной частью).

Разрядную сетку чисел в формате с фиксированной точкой, можно представить, как показано на рисунке 1.13.

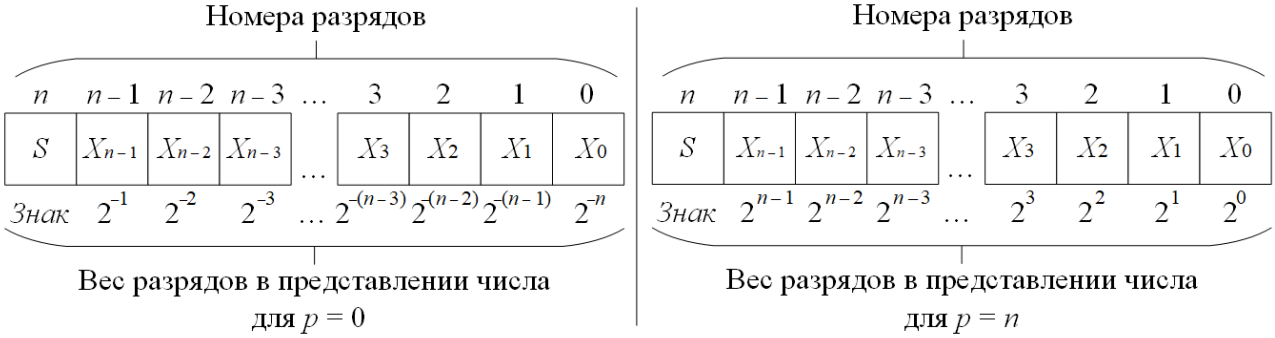


Рисунок 1.13 – Разрядная сетка чисел в формате с фиксированной точкой

Тогда диапазоны представления таких n-разрядных чисел имеют вид:

$$\text{а) для } p = 0 \quad \begin{cases} |x|_{\max} = 0{,}111 \ldots 11 = 1 - 2^{-n}; \\ |x|_{\min} = 0{,}000 \ldots 01 = 2^{-n}; \\ 1 - 2^{-n} \geq |x| \geq 2^{-n}; \end{cases} \qquad (1.26)$$

$$\text{б) для } p = n \quad \begin{cases} |x|_{\max} = 111 \ldots 11 = 2^n - 1; \\ |x|_{\min} = 000 \ldots 01 = 1; \\ 2^n - 1 \geq |x| \geq 1. \end{cases} \tag{1.27}$$

Кодирование вещественных числе в формате с фиксированной точкой происходит аналогично кодированию целых чисел.

Прямой базовый код

При кодировании числа в прямой базовый код запись вещественного числа (правильной дроби) в формате с фиксированной точкой B формируется по следующему правилу:

$$[B]_{\text{ПК}} = \begin{cases} 0.B, \text{если } B \geq 0\ ; \\ 1.|B|, \text{если } B < 0\ . \end{cases} \tag{1.28}$$

Обратный базовый код

При кодировании числа в обратный базовый код запись дробного числа (правильной дроби) B формируется по следующему правилу:

$$[B]_{\text{ОК}} = \begin{cases} 0.B, \text{если } B \geq 0; \\ 1.((1 - q^n) + B), \text{если } B < 0, \end{cases} \tag{1.29}$$

где $(1 - q^n)$ – максимальная включенная граница диапазона изменения представляемых чисел.

Дополнительный базовый код

При кодировании числа в дополнительный базовый код запись дробного числа (правильной дроби) B формируется по следующему правилу:

$$[B]_{\text{ДК}} = \begin{cases} 0.B, \text{если } B \geq 0\ ; \\ 1.(1 + B), \text{если } B < 0\ , \end{cases} \tag{1.30}$$

где 1 – максимальная невключенная граница диапазона изменения представляемых чисел.

Аналогично с целыми числами при выполнении арифметических операций используются *модифицированные коды*.

Одним из важнейших параметров представления вещественных чисел является ошибка представления, т. е. точность. Ошибка представления может быть абсолютной (Δ) или относительной (σ). Для формата с фиксированной точкой максимальные значения этих ошибок определяются следующим образом:

$$\Delta_{\max} = 0{,}5 \cdot 2^{-n} = 2^{-(n+1)}; \tag{1.31}$$

$$\sigma_{\max} = \frac{\Delta_{\max}}{A_{\min}} = \frac{2^{-(n+1)}}{2^{-n}}, \qquad (1.32)$$

где $A_{\min}$ – минимальное отличное от нуля значение числа.

Таким образом, в худшем случае относительная ошибка может достигать сравнительно большого значения – 50 %.

Достоинства формата представления вещественных чисел с фиксированной точкой (запятой):

– требуются сравнительно несложные операционные арифметические устройства с высоким быстродействием;

– операции сложения и вычитания выполняются как с целыми знаковыми числами;

– поддержка старых процессоров, микроконтроллеров и т. п.

В настоящее время формат представления вещественных чисел с фиксированной точкой (запятой) почти не используется из-за ряда недостатков:

– в таком формате существующими в языках программирования вещественными типами могут храниться сравнительно небольшие значения;

– происходит быстрый сдвиг точки (запятой) при умножении;

– необходимость предварительного масштабирования исходных чисел и необходимость слежения за положением точки (запятой) в ходе выполнения вычислений;

– относительно невысокая точность представляемых чисел.

Вещественное число в формате с плавающей точкой (запятой) в общем случае представляет собой смешанную дробь и имеет структуру, представленную на рисунке 1.14.

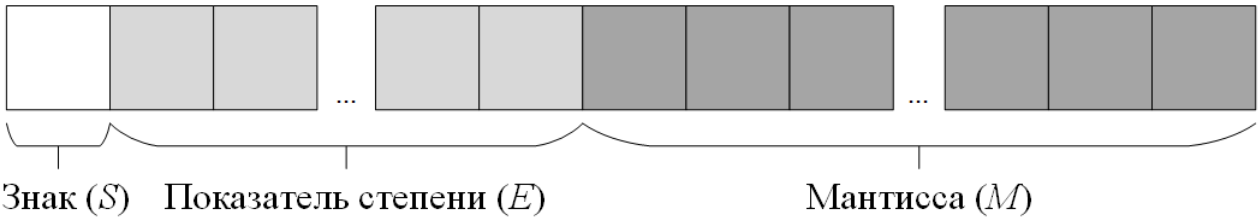


Рисунок 1.14 – Структура представления числа в формате с плавающей точкой

Местоположение условной точки в записи числа может быть различным, а т. к. сама точка в записи числа не присутствует, то для однозначного задания числа необходима не только его запись, но и информация о том, где в записи числа располагается условная точка, отделяющая целую и дробную части.

При представлении в формате с плавающей точкой (запятой) число Q формируется из трех частей:

– знак числа (S);

– показатель степени числа (E – экспонента), отображающий местоположение условной точки в записи числа;

– мантисса (M), отображающая непосредственно запись самого числа.

При этом показатель степени числа представлен целым числом в формате с фиксированной точкой (запятой), а мантисса представлена в виде правильной дроби в формате с фиксированной точкой (запятой).

Таким образом, целая часть числа должна быть равна единице, которая явно не фигурирует в записи (записывается только дробная часть).

Количественная оценка числа определяется как

$$Q = (-1)^S \cdot q^E \cdot M, \tag{1.33}$$

где S – знак всего числа (для положительных нуль, а для отрицательных единица);

q – основание системы счисления;

E – показатель степени числа;

M – мантисса числа.

Максимальная относительная ошибка σ_{max} представления чисел в формате с плавающей точкой определяется как

$$\sigma_{\max} = \frac{\Delta_{\max}}{A_{\min}} = \frac{2^{-(n+1)} \cdot 2^{2(2k-1)}}{x_{M\min} \cdot 2^{2(2k-1)}} = \frac{2^{-(n+1)} \cdot 2^{2(2k-1)}}{2^{-1} \cdot 2^{2(2k-1)}} = \frac{2^{-(n+1)}}{2^{-1}} = 2^{-n}, \tag{1.34}$$

где n – количество разрядов в записи модуля мантиссы числа;

k – количество разрядов в записи показателя степени числа.

Относительная ошибка при представлении чисел в формате с плавающей точкой существенно меньше, чем в случае с фиксированной точкой. Это, а также больший диапазон изменения представляемых чисел является основным преимуществом представления чисел с плавающей точкой.

В настоящее время представление и обработка вещественных чисел в формате с плавающей точкой (запятой) регламентируется международным стандартом *IEEE Std* 754–2019 (*IEEE Standard for Floating-Point Arithmetic* 754–2019).

Стандарт *IEEE Std* 754–2019 регламентирует следующие вопросы:

– представление нормализованных и денормализованных (субнормализованных) вещественных чисел с плавающей точкой;

– форматы вещественных чисел (к ним, по сути, приравниваются типы вещественных данных в типизированных языках программирования высокого уровня);

– смещение показателя степени для разных форматов вещественных чисел с плавающей точкой;

– представление специальных величин (±0, ±∞, «не числа» (*NaN/NaNs*));

– правила округления вещественных чисел с плавающей точкой;

– точность представления вещественных чисел с плавающей точкой.

Нормальная и нормализованная форма вещественных чисел с плавающей точкой

Нормальная форма представления вещественного числа с плавающей точкой – это такая форма представления числа, при которой мантисса находится в интервале от нуля включительно до единицы не включительно, т. е. [0; 1).

Нормальная форма представления вещественного числа с плавающей точкой имеет существенный недостаток: одно и то же число может быть представлено несколькими вариантами, т. е. неоднозначно.

Например, десятичное число 0,001 можно записать множеством вариантов: $0{,}001 \cdot 10^0$, или $0{,}01 \cdot 10^{-1}$, или $0{,}1 \cdot 10^{-2}$, или $0{,}0001 \cdot 10^1$, или $0{,}00001 \cdot 10^2$, или $0{,}000001 \cdot 10^3$ и т. д.

В соответствии со стандартом *IEEE Std* 754–2019 вещественные числа с плавающей точкой должны быть представлены в нормализованной форме.

Нормализованная форма представления вещественного числа с плавающей точкой подразумевает, что мантисса десятичного числа принимает значения от 1 (включительно) до 10 (не включительно), а мантисса двоичного числа принимает значения от 1 (включительно) до 2 (не включительно) – т. е. может быть только единицей. Таким образом, в мантиссе слева от запятой (целая часть) до применения порядка должен находиться только один разряд, причем единичный. И поскольку старший двоичный разряд (целая часть) мантиссы вещественного числа с плавающей точкой в нормализованном виде всегда равен «1», то стандарт *IEEE Std* 754–2019 предписывает не записывать его (справедливо только для двоичных чисел), сэкономив таким образом один бит.

Для двоичной системы счисления нормализованная мантисса должна иметь в старшем разряде модуля прямого кода значение, равное единице, т. е. для двоичной системы счисления мантисса должна удовлетворять неравенству

$$1 > |M| \geq 0{,}5. \qquad (1.35)$$

В нормализованной форме любое число представляется единственным образом. Исключения: ±0, ±∞ и неопределенность («не число», *NaN/NaNs*).

Пример нормализации

Нормальная форма записи вещественных чисел с плавающей точкой:

$5{,}0_{10} = 101{,}0_2 = 101{,}0e0 = 10{,}1e1 = 1{,}01e2 = 0{,}101e3 = 1010{,}0e(-1)$;

$2{,}5_{10} = 10{,}1_2 = 10{,}1e0 = 1{,}01e1 = 0{,}101e2 = 101{,}0e(-1)$.

Нормализованная форма записи вещественных чисел с плавающей точкой:

$5{,}0_{10} = 1{,}01e2$;

$2{,}5_{10} = 1{,}01e1$.

Перемещая плавающую точку (запятую) на n разрядов влево, необходимо увеличить показатель степени E на значение n, а перемещая плавающую точку (запятую) на n разрядов вправо, необходимо уменьшить показатель степени E на значение n. При этом плавающая точка (запятая) двигается влево или вправо так, чтобы в целой части был только один ненулевой разряд (единичный).

Денормализованная (субнормализованная) форма вещественных чисел с плавающей точкой

Денормализация – это способ повышения точности представления чисел, т. е. увеличение количества представимых значений в окрестности нуля для вещественных чисел с плавающей точкой.

Таким образом, каждое значение денормализованного числа меньше самого маленького нормализованного значения числа.

Отличия представления нормализованного и денормализованного чисел представлены ниже.

$$\text{Нормализованное число: } (-1)^S \cdot 1, M \cdot 2^E, \text{если } E_{\min} \le E \le E_{\max}. \quad (1.36)$$

$$\text{Денормализованное число: } (-1)^S \cdot 0, M \cdot 2^{E_{\min}}, \text{если } E = E_{\min} - 1. \quad (1.37)$$

Денормализованные числа позволяют более точно обрабатывать очень маленькие значения, однако, ввиду сложности реализации на аппаратном уровне, такая обработка чаще всего реализуется программно и выполняется в десятки раз медленнее, чем обработка нормализованных чисел.

Значение мантиссы денормализованных чисел лежит в диапазоне $0{,}1 \le M < 1$.

Формула для расчета денормализованых чисел может быть записана как

$$A_{\text{денорм}} = (-1)^S \cdot 2^{(E-(2^{(R-1)}+2))} \cdot \left(\frac{M}{2^m}\right), \quad (1.38)$$

где $A_{\text{денорм}}$ – представление денормализованного числа;

S – знак всего числа (для положительных – нуль, а для отрицательных – единица);

E – показатель степени со смещением;

$(2^{(R-1)} + 2)$ – заданное смещение для денормализации;

R – количество разрядов для представления показателя степени E;

M – дробная часть мантиссы;

m – количество разрядов для представления мантиссы M.

Форматы представления вещественных чисел с плавающей точкой. Смещение показателя степени для разных форматов

Стандарт *IEEE Std* 754–2019 определяет следующие двоичные форматы представления вещественных чисел с плавающей точкой:

– формат половинной точности (*Binary16*, *Half Precision*);

– формат одинарной точности (*Binary32*, *Single Precision*, *float*);

– формат двойной точности (*Binary64*, *Double Precision*, *double*);

– формат четверной точности (*Binary128*, *Quadruple Precision*).

Форматы определяют следующие характеристики вещественных чисел:

– размер выделяемой памяти под хранения значения числа;

– точность представления вещественных чисел (число разрядов дробной части числа);

– количество разрядов для записи показателя степени;

– количество разрядов для записи мантиссы;

– допустимый диапазон изменения значений чисел;

– значение смещения показателя степени для возможности представления знаковых значений.

Наглядное сравнение характеристик форматов при представлении знаковых вещественных чисел с плавающей точкой на 32-разрядной вычислительной машине представлено в таблице 1.9.

Таблица 1.9 – Характеристики форматов представления вещественных чисел с плавающей точкой

Формат	Размер выделяемой памяти, байт	Число бит под показатель степени *E*	Число бит под мантиссу *M*	Смещение показателя степени	Диапазон значений[1]	Точность (десятичных цифр дробной части)
Половинный	2	5	10	15	$-6{,}10 \cdot 10^{-5}...+65504$	4
Одинарный (*float*[2])	4	8	23	127	$-3{,}4 \cdot 10^{38}..+3{,}4 \cdot 10^{38}$	7
Двойной (*double*[2])	8	11	52	1023	$-1{,}7 \cdot 10^{308}..+1{,}7 \cdot 10^{308}$	15
Четверной (*long double*[3])	16	15	112	16383	$-3{,}4 \cdot 10^{4932}…+3{,}4 \cdot 10^{4932}$	19

Примечания

[1] Диапазон значений только примерно указывает границы допустимых значений без учета возрастающей погрешности, с ростом абсолютного значения, и существования денормализованных чисел.

[2] Соответствие типам данных в языках программирования *C/C++*.

[3] Соответствие типу данных с модификатором для некоторых архитектур, например, в сопроцессоре *Intel*.

Рассмотрим данные характеристики подробнее на примере формата одинарной точности (типа данных *Single* в языке программирования *Delphi* или *float* в языках программирования *C/C++).*

Под хранение значений вещественных чисел с плавающей точкой в формате одинарной точности выделяется 4 байта памяти, т. е. 4 байта · 8 = 32 бита. Из которых:

– под представление значения мантиссы выделяется 23 бита (с 0-го по 22-й бит);

– под представление значения показателя степени выделяется 8 бит (с 23-го по 30-й бит);

– под представление знака всего числа *S* выделяется 1 бит (старший, 31-й);

– целая часть мантиссы нормализованных чисел, равная единице, никуда не записывается, а только условно подразумевается.

Наглядно формат одинарной точности представлен на рисунке 1.15.

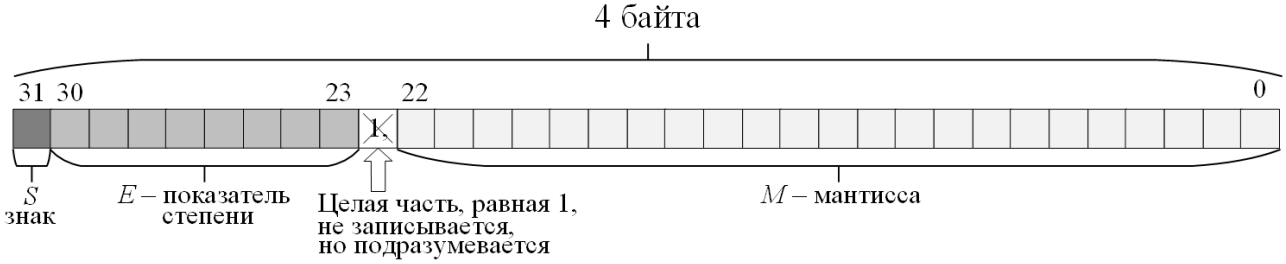


Рисунок 1.15 – Структура формата одинарной точности

Поскольку старший (31-й) бит хранит знак всего числа, т. е. по сути является знаком мантиссы, то, как видно из рисунка 1.15, нигде не предусмотрен знак показателя степени, который может быть как положительный, так и отрицательный, в зависимости от того, каким получится число после нормализации.

Для того чтобы учесть знак показателя степени, стандарт *IEEE Std* 754–2019 вводит понятие смещения показателя степени. В примере формата одинарной точности под представление порядка степени выделяется 8 бит. Восемь двоичных разрядов (бит) могут представлять $2^8 = 256$ уникальных двоичных комбинаций, эквивалентных в десятичной системе счисления множеству [0...255]. Разделив это множество поровну на отрицательную и положительную области, с учетом того, что нуль также является неотрицательным числом, получим, что значение 127 является серединой данного множества (т. е. нулем). Тогда двоичные комбинации (00000000...01111110), соответствующие десятичному подмножеству от 0 до 126, будут соответствовать области отрицательных значений показателя степени, двоичная комбинация (01111111), соответствующая десятичному значению 127, будет являться началом положительной области (нулевым значением показателя степени), а двоичные комбинации (10000000...11111111), соответствующие десятичному подмножеству от 128 до 255, будут принадлежать области положительных ненулевых значений показателя степени. Наглядно такое смещение для формата одинарной точности показано в таблице 1.10.

Аналогичным образом рассчитываются смещения и диапазоны для других форматов точности представления вещественных чисел с плавающей точкой.

Таблица 1.10 – Диапазоны отрицательных, нулевого и положительных значений показателя степени для формата одинарной точности представления вещественных чисел с плавающей точкой

Распределение 8 бит, отводимых под представление порядка степени E ($2^8 = 256$ комбинаций в диапазоне [0...255])		Значение, записанное в показателе степени E, с учетом смещения		Фактическое значение показателя степени без смещения
		Десятичная система счисления	*Двоичная система счисления*	*Десятичная система счисления*
Область отрицательных значений показателя степени	min	0	00000000	–127
	max	126	01111110	–1
Нулевое значение показателя степени		127	01111111	0
Область положительных ненулевых значений показателя степени	min	128	10000000	+1
	max	255	11111111	+127
Примечание – Значение показателя степени указано теоретически, без учета того, что некоторые комбинации зарезервированы для представления неопределенности («не числа» (*NaN*/*NaNs*)), ±∞ и ±0.				

Пример представления числа с положительной степенью

Представим десятичное число $+155{,}625_{10}$ в формате одинарной точности по стандарту *IEEE Std* 754–2019.

Поскольку заданное число представляет собой неправильную дробь, отдельно переведем в двоичную систему целую и дробные части:

$+155_{10} = +10011011_2$;

$+0{,}625_{10} = +0{,}101_2$.

Таким образом,

$+155{,}625_{10} = +10011011{,}101_2$.

Обратная проверка:

$10011011{,}101_2 = 1 \cdot 2^7 + 0 \cdot 2^6 + 0 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 + + 1 \cdot 2^{-1} + 0 \cdot 2^{-2} + 1 \cdot 2^{-3} = 128 + 0 + 0 + 16 + 8 + 0 + 2 + 1 + 0{,}5 + 0 + 0{,}125 = = 155{,}625_{10}$.

Для наглядности приведем полученное число к нормализованному виду в десятичной системе счисления:

$+155{,}625e0 = +1{,}55625e2$ (поскольку мы переместили плавающую точку на два разряда влево, то увеличиваем показатель степени на значение, равное двум).

Приведем полученное число к нормализованному виду в двоичной системе счисления:

$10011011{,}101e0 = 1{,}0011011101e7$ (поскольку мы переместили плавающую точку на семь разрядов влево, то увеличиваем показатель степени на значение, равное семи).

Таким образом, имеем мантиссу $M = 1{,}0011011101$ и показатель степени $E = 7$.

Выполним корректировку мантиссы смещением на значение 127 (для одинарной точности по условию):

$127 + e = 127 + 7 = 134_{10} = 10000110_2$.

Поскольку заданное число положительное, то старший (31-й) знаковый бит *S* будет нулевым.

Далее слева направо (с 30-го по 23-й) бит будет записано значение показателя степени со смещением *E*, равное 10000110.

Далее слева направо (с 22-го по 13-й биты, т. е. в старшие разряды мантиссы) будет записана дробная часть мантиссы *M*, т. к. целая часть (единица) не записывается с целью экономии бита, а подразумевается (поскольку число нормализованное, то в целой части двоичного числа всегда будет единица), при этом младшие 13 бит (с 0-го по 12-й) заполняются нулями, поскольку для дробной части младшие нулевые разряды являются незначащими в любой системе счисления.

Результат представления числа $+155{,}625_{10}$ в формате одинарной точности по стандарту *IEEE Std* 754–2019 проиллюстрирован на рисунке 1.16.

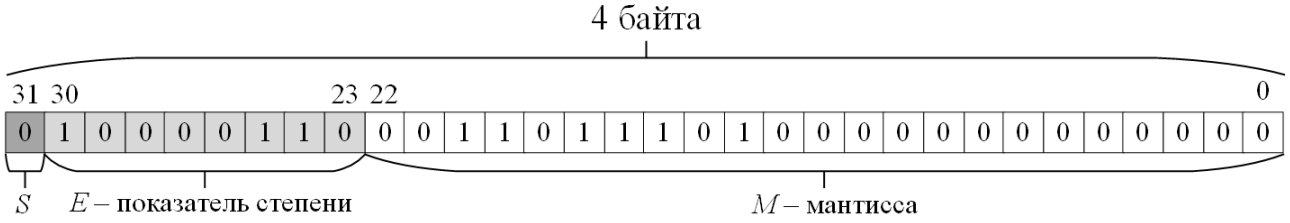


Рисунок 1.16 – Результат представления числа $+155{,}625_{10}$

Пример представления числа с отрицательной степенью

Представим десятичное число $-0{,}125_{10}$ в формате одинарной точности по стандарту *IEEE Std* 754–2019.

Переведем в двоичную систему дробную часть:

$-0{,}125_{10} = -0{,}001_2$.

Обратная проверка:

$0{,}001_2 = 0 \cdot 2^{0} + 0 \cdot 2^{-1} + 0 \cdot 2^{-2} + 1 \cdot 2^{-3} = 0 + 0 + 0 + 0{,}125 = 0{,}125_{10}$.

Для наглядности приведем полученное число к нормализованному виду в десятичной системе счисления:

$0{,}125e0 = 1{,}25e(-1)$ (поскольку мы переместили плавающую точку на один разряд вправо, то уменьшаем показатель степени на значение, равное единице).

Приведем полученное число к нормализованному виду в двоичной системе счисления:

$0{,}001e0 = 1{,}0e(-3)$ (поскольку мы переместили плавающую точку на три разряда вправо, то уменьшаем показатель степени на значение, равное трем).

Таким образом, имеем мантиссу $M = 1{,}0$ и показатель степени $E = -3$.

Выполним корректировку мантиссы смещением на значение 127 (для одинарной точности по условию):

$127 + e = 127 + (-3) = 124_{10} = 01111100_2$.

Поскольку заданное число отрицательное, то старший (31-й) знаковый бит *S* будет единичным.

Далее слева направо (с 30-го по 23-й) бит будет записано значение показателя степени со смещением *E*, равное 01111100.

Далее слева направо (с 22-го по 0-й биты, т. е. в старшие разряды мантиссы) будет записана дробная часть мантиссы *M*, т. к. целая часть (единица) не записывается с целью экономии бита, а подразумевается (поскольку число нормализованное, то в целой части двоичного числа всегда будет единица), причем в данном случае дробная часть мантиссы равна нулю.

Результат представления числа $-0{,}125_{10}$ в формате одинарной точности по стандарту *IEEE Std* 754–2019 представлен на рисунке 1.17.

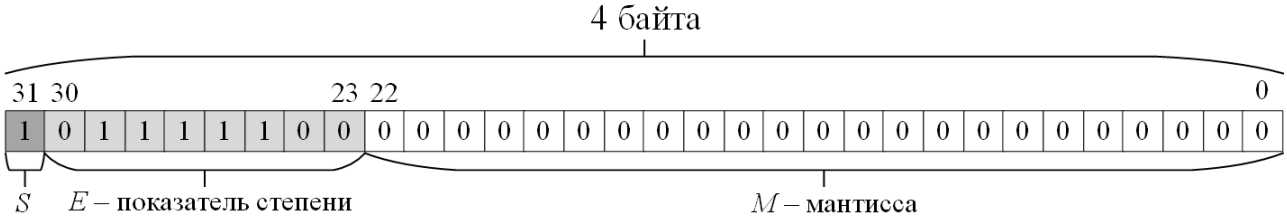


Рисунок 1.17 – Результат представления числа $-0{,}125_{10}$

Пример перевода вещественного числа с плавающей точкой в десятичное

Для обратного перевода числа, представленного по стандарту *IEEE Std* 754–2019, к десятичному представлению можно воспользоваться формулой

$$A_{10} = (-1)^S \cdot 2^{(E-(2^{(R-1)}-1))} \cdot \left(1 + \frac{M}{2^m}\right), \tag{1.39}$$

где A_{10} – представление числа в десятичной системе счисления;

S – знак числа (нуль для положительных, единица для отрицательных);

E – показатель степени со смещением;

$(2^{(R-1)} - 1)$ – заданное смещение в соответствии с форматом;

R – количество разрядов для представления показателя степени E;

M – дробная часть мантиссы;

m – количество разрядов для представления мантиссы M.

Переведем обратно в десятичную систему счисления число, представленное на рисунке 1.16 в формате одинарной точности:

$S = 0$;

$E = 10000110_2 = 134_{10}$;

$M = 00110111010000000000000_2 = 1810432_{10}$;

$R = 8$;

$m = 23$.

Подставив значения в (1.39), получим

$$A_{10} = (-1)^0 \cdot 2^{(134-2^{(8-1)}-1)} \cdot \left(1 + \frac{1810432}{2^{23}}\right) =$$
$$= 1 \cdot 2^7 \cdot (1 + 0{,}2158203125) = 128 \cdot 1{,}2158203125 = 155{,}625.$$

В результате перевода мы получили верное значение в десятичной системе счисления, равное $A_{10} = +155{,}625_{10}$.

Представление особых значений чисел с плавающей точкой

К особым значениям вещественного числа с плавающей точкой относятся:

– плюс нуль;

– минус нуль;

– плюс бесконечность;

– минус бесконечность;

– неопределенность («не число» (*NaN*/*NaNs*)).

Нуль (со знаком)

В нормализованной форме вещественное число с плавающей точкой невозможно представить как однозначный нуль. Для представления нуля зарезервированы специальные комбинации мантиссы и показателя степени.

Число считается нулем, если все его биты (и мантиссы, и показателя степени), кроме знакового, равны нулю. При этом в зависимости от значения бита знака нуль может быть как положительным, так и отрицательным (рисунок 1.18).

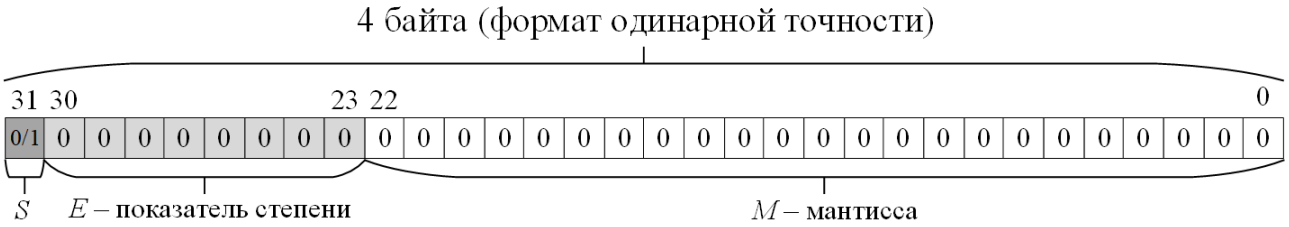


Рисунок 1.18 – Представление нуля по стандарту *IEEE Std* 754–2019

Кроме того, для нуля со знаком справедливы следующие правила арифметики:

1) $\frac{-0}{|x|} = -0, \text{если } x \neq 0;$

2) (–0) · (–0) = +0;

3) |*x*| · (–0) = –0;

4) *x* ±0 = *x*;

5) (–0) + (–0) = –0;

6) (+0) + (+0) = +0;

7) $\frac{-0}{-\infty} = +0;$

8) $\frac{|x|}{-0} = -\infty, \text{если } x \neq 0.$

Бесконечность (со знаком)

Вещественное число в формате с плавающей точкой считается равным бесконечности, если все двоичные разряды его показателя степени равны единице, а мантисса равна нулю. Знак бесконечности определяется знаковым битом числа (рисунок 1.19).

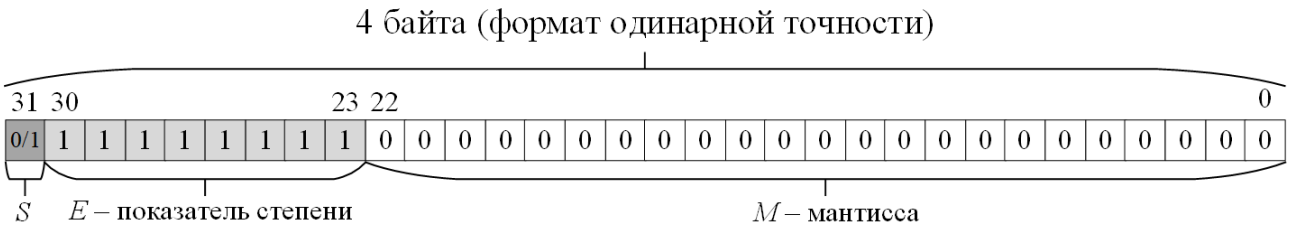


Рисунок 1.19 – Представление бесконечности по стандарту *IEEE Std* 754–2019

Получить бесконечность можно при переполнении и при делении ненулевого числа на нуль. При этом

$$\frac{x}{0} = \begin{cases} +\infty, & \text{если } x > 0; \\ NaN, & \text{если } x = 0; \\ -\infty, & \text{если } x < 0. \end{cases}$$

Неопределенность («не число» / «не числа» (*NaN/NaNs*))

Неопределенность является результатом арифметических операций, если во время их выполнения произошла ошибка. Неопределенность также называется «не числом» (от англ. «*Not a Number*», *NaN*). В стандарте *IEEE Std* 754–2019 представлена как число, в котором все двоичные разряды показателя степени единичные, а мантисса не равна нулю, т. е. могут быть различные комбинации разрядов, но все разряды мантиссы не должны быть нулевыми (рисунок 1.20).

Исходя из этого

«*одна неопределенность ≠ другая неопределенность*».

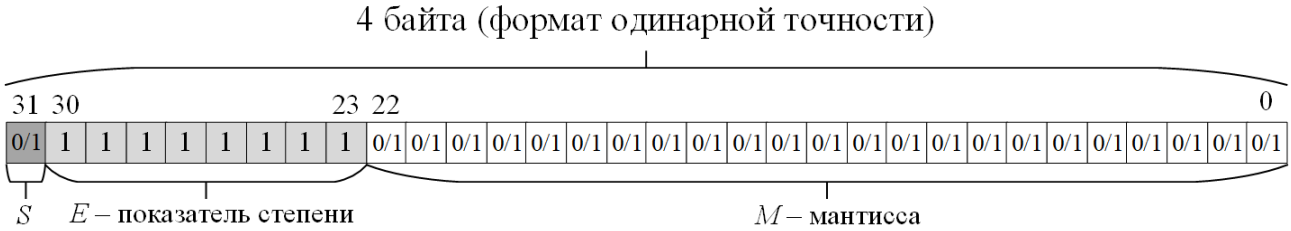


Рисунок 1.20 – Неопределенность (*NaN/NaNs*) по стандарту *IEEE Std* 754–2019

Неопределенность («не число» / «не числа», *NaN/NaNs*) можно получить в результате следующих операций:

1) $+\infty + (-\infty) = NaN$;

2) $0 \cdot \infty = NaN$;

3) $\frac{\pm 0}{\pm 0} = NaN$;

4) $\frac{\pm \infty}{\pm \infty} = NaN$;

5) $\sqrt{x} = NaN$, если $x < 0$ и др.

Таким образом, полный диапазон представления вещественных чисел с плавающей точкой в формате с одинарной точностью показан на рисунке 1.21.

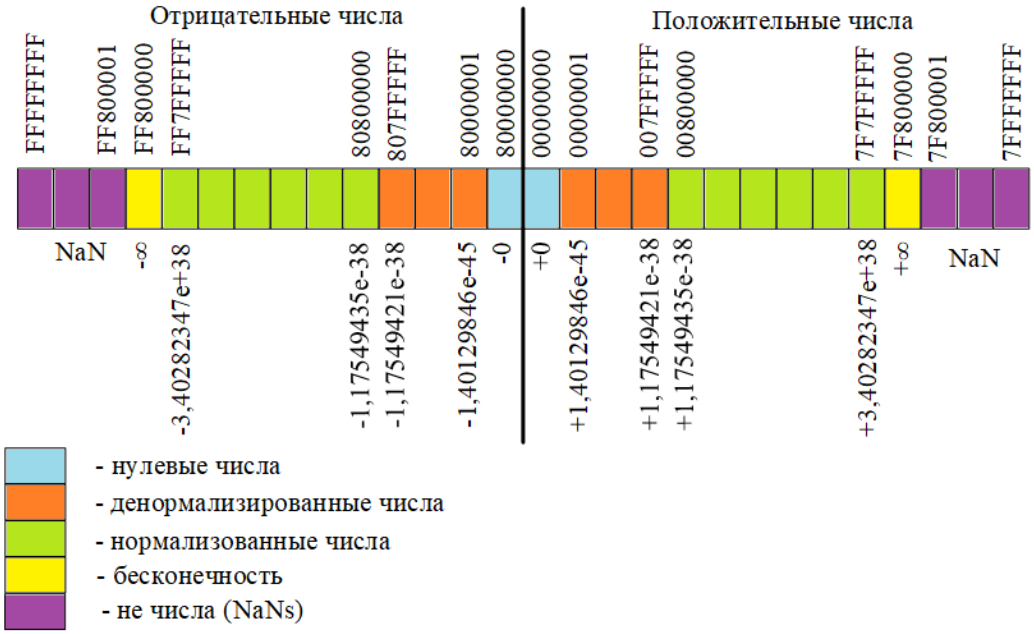


Рисунок 1.21 – Диапазон представления чисел в формате с одинарной точностью

Полный диапазон представления вещественных чисел с плавающей точкой в формате с двойной точностью представлен на рисунке 1.22.

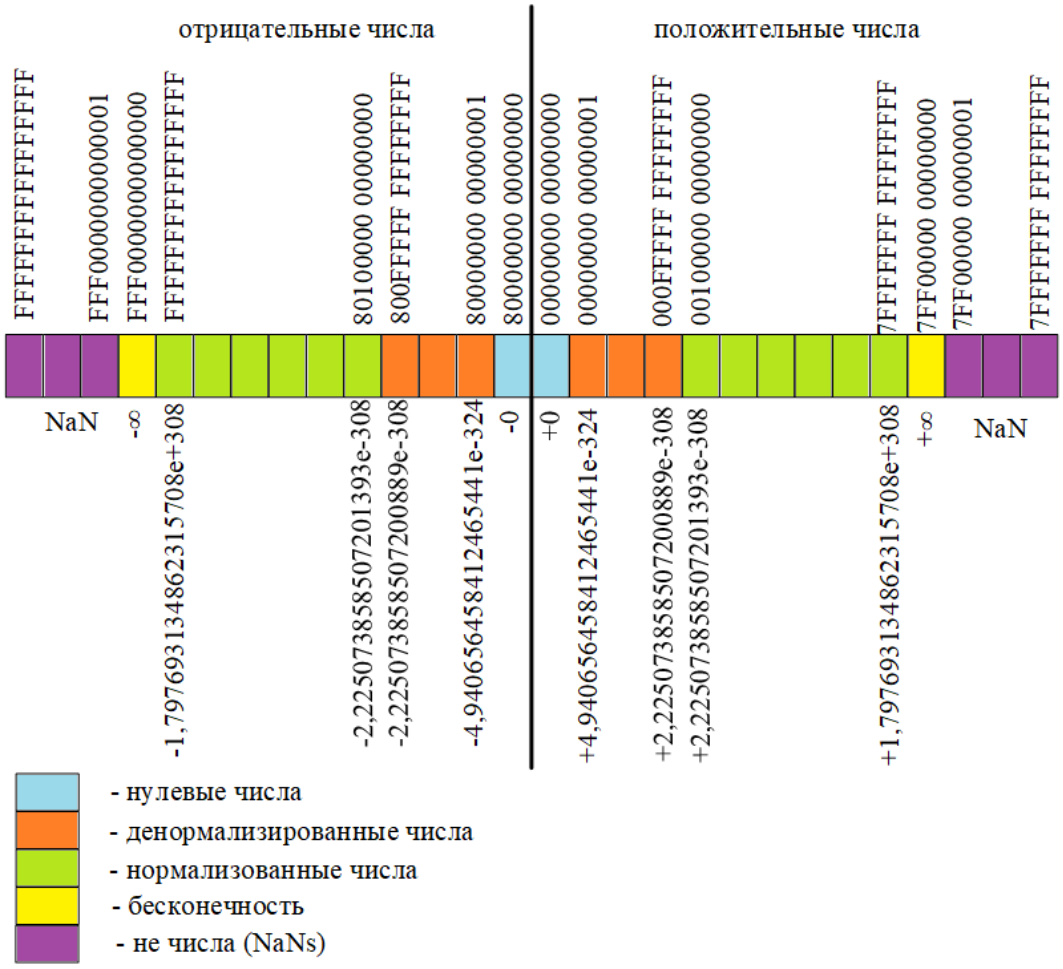


Рисунок 1.22 – Диапазон представления чисел в формате с двойной точностью

На рисунке 1.23 отражены представления наименьших и наибольших нормализованных положительных и отрицательных вещественных чисел с плавающей точкой.

Наименьшее положительное

Знак	Смещенный порядок					Модуль мантиссы			
0	0	0	...	0	1	0	0	...	0

Наибольшее отрицательное

Знак	Смещенный порядок					Модуль мантиссы			
1	0	0	...	0	1	0	0	...	0

Наибольшее положительное

Знак	Смещенный порядок					Модуль мантиссы			
0	1	1	...	1	0	1	1	...	1

Наименьшее отрицательное

Знак	Смещенный порядок					Модуль мантиссы			
1	1	1	...	1	0	1	1	...	1

Рисунок 1.23 – Представления знаковых наименьших и наибольших вещественных чисел с плавающей точкой

Точность представления вещественных чисел с плавающей точкой

Абсолютная максимальная погрешность для вещественных чисел с плавающей точкой будет равна в пределе половине шага чисел. При этом шаг чисел будет удваиваться с увеличением показателя степени двоичного числа на одну единицу. Таким образом, чем дальше от нуля, тем шире будет шаг чисел по числовой оси и тем больше максимальная ошибка представления.

Наглядно это представлено на рисунке 1.24.

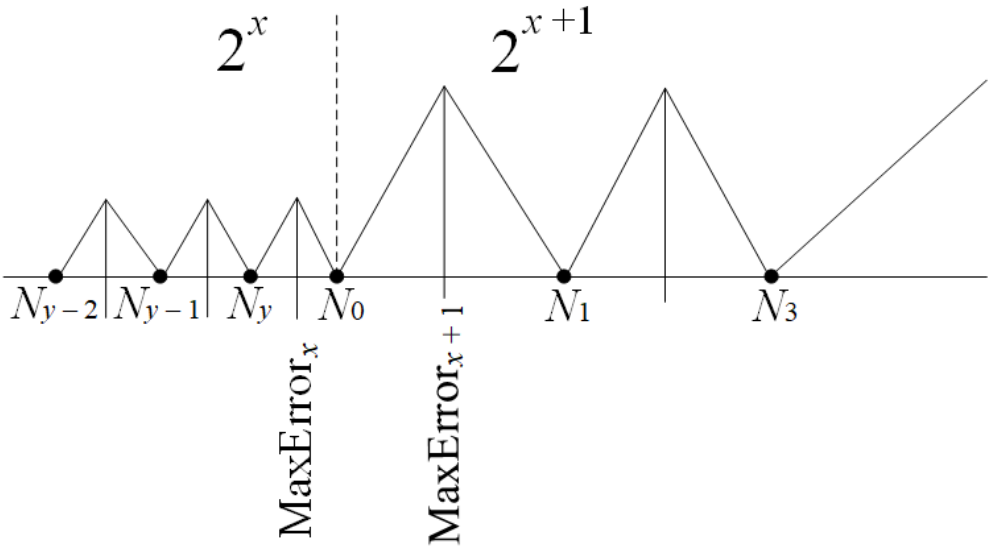


Рисунок 1.24 – Зависимость максимальной абсолютной ошибки от шага чисел и показателя степени

Характеристики ошибок представления нормализованных и денормализованных вещественных чисел с плавающей точкой представлены в таблице 1.11.

Таблица 1.11 – Характеристики ошибок представления вещественных чисел с плавающей точкой

Показатель	Нормализованные числа		Денормализованные числа	
	одинарной точности	двойной точности	одинарной точности	двойной точности
Шаг чисел h	2^{E-150}	2^{E-1075}	2^{E-149}	2^{E-1074}
Предел макс. абсолютной ошибки	$\lim\frac{h}{2} = 2^{E-151}$	$\lim\frac{h}{2} = 2^{E-1076}$	$\lim\frac{h}{2} = 2^{E-150}$	$\lim\frac{h}{2} = 2^{E-1075}$
Относительная ошибка	$\frac{2^{E-151}}{A_{\text{нормал}}} \cdot 100\ \%$	$\frac{2^{E-1076}}{A_{\text{нормал}}} \cdot 100\ \%$	$\frac{2^{E-150}}{A_{\text{денормал}}} \cdot 100\ \%$	$\frac{2^{E-1075}}{A_{\text{денормал}}} \cdot 100\ \%$
Макс. относительная ошибка	$\frac{1}{2^{24}+2M} = 2^{-23} \cdot 100\ \%$	$\frac{1}{2^{53}+2M} = 2^{-52} \cdot 100\ \%$	$\frac{1}{2M}$	$\frac{1}{2M}$

Из таблицы 1.11 следует, что основная масса вещественных чисел с плавающей точкой имеет стабильную небольшую относительную погрешность.

Максимально возможная относительная погрешность для числа одинарной точности составит $2^{-23} \cdot 100\ \% = 11{,}920928955078125e(-6)\ \%$.

Максимально возможная относительная погрешность для числа двойной точности составит $2^{-52} \cdot 100\% = 2{,}220446049250313080847263336181 6e(-14)\ \%$.

Округление вещественных чисел с плавающей точкой

Стандарт *IEEE Std* 754–2019 предусматривает четыре способа округления вещественных чисел с плавающей точкой:

– округление, стремящееся к ближайшему целому;

– округление, стремящееся к нулю;

– округление, стремящееся к +∞;

– округление, стремящееся к –∞.

По умолчанию используется округление к ближайшему целому, однако довольно часто используется и округление, стремящееся к нулю, поскольку при таком округлении необходимо просто отбросить незначащие разряды числа, поэтому он самый легкий с точки зрения аппаратной реализации. Способы округления продемонстрированы в таблице 1.12.

Таблица 1.12 – Примеры округления различными способами

Исходное десятичное число	Способ округления			
	К ближайшему целому	К нулю	К –∞	К +∞
1,33	1,3	1,3	1,4	1,3
–1,33	–1,3	–1,3	–1,3	–1,4
1,37	1,4	1,3	1,4	1,3
–1,37	–1,4	–1,3	–1,3	–1,4
1,35	1,4	1,3	1,4	1,3
–1,35	–1,4	–1,3	–1,3	–1,4

1.5 Контрольные вопросы

1 Как классифицируются системы счисления? Назовите примеры систем счисления для разных их видов.

2 Назовите и поясните основные критерии выбора системы счисления.

3 Какая система счисления является наилучшей по критерию экономичности и почему?

4 Какие методы перевода из заданной системы счисления в искомую являются универсальными и почему?

5 Чем отличается перевод целых и перевод дробных чисел из заданной системы счисления в искомую методом с использованием ве́сов разрядов?

6 Как формируется запись числа в искомой системе счисления при преобразовании методом деления (умножения) на новое основание?

7 Какие условия должны выполняться для использования метода особого соотношения оснований систем счисления?

8 Когда целесообразно использовать метод особого соотношения оснований систем счисления, если не выполняются необходимые для этого условия?

9 В каких случаях применяется схема Горнера?

10 Как формируется запись положительного алгебраического числа в обратном и дополнительном кодах?

11 Как формируется запись отрицательного алгебраического числа в обратном и дополнительном кодах?

12 Чем отличаются модифицированные коды от базовых и в чем их преимущество?

13 Как определяется переполнение модульного поля в модифицированных кодах?

14 Какой разряд знакового поля является знакоопределяющим в модифицированных кодах?

15 В каких форматах представляются вещественные числа?

16 Какой стандарт регламентирует представление вещественных чисел в формате с плавающей точкой (запятой)?

17 Какие форматы точности представления вещественных чисел в формате с плавающей точкой (запятой) определяет стандарт *IEEE Std* 754–2019?

18 Каким образом учитывается знак вещественного числа в формате с плавающей точкой (запятой) в памяти компьютера?

19 Каким образом учитывается знак показателя степени при представлении вещественного числа в формате с плавающей точкой (запятой) в памяти вычислительной машины?

20 Чему равно смещение показателя степени для формата одинарной точности в соответствии со стандартом *IEEE Std* 754–2019?

21 Что такое нормализованные вещественные числа?

22 Какие виды округления вещественных чисел определяет стандарт *IEEE Std* 754–2019?

2 АРИФМЕТИЧЕСКИЕ ОСНОВЫ ЭЛЕКТРОННЫХ ВЫЧИСЛИТЕЛЬНЫХ МАШИН

В настоящее время практически не осталось аналоговой техники. Все устройства, от бытовых приборов до космических аппаратов, являются цифровыми. Как известно, цифровые устройства, в том числе компьютерная техника, при обработке информации используют в основном двоичную систему счисления. Именно поэтому в большей своей части компьютерная арифметика является двоичной арифметикой. Это обусловлено двумя основными причинами.

Во-первых, алгоритмы арифметических операций двоичной арифметики (т. е. арифметики, использующей двоичную позиционную систему) очень просты и являются в определенном смысле простейшими среди подобных алгоритмов для всех позиционных числовых систем. Во-вторых, дискретные (не аналоговые) электронные схемы, как самые современные, так и использовавшиеся много лет назад, имеют в определенном смысле двоичную природу и легко описываются на языке алгебры логики.

В двоичной системе счисления арифметические операции (сложение, вычитание, умножение и деление) выполняются по тем же правилам, что и в привычной нам десятичной системе счисления, т. к. они обе являются позиционными (наряду с восьмеричной, шестнадцатеричной и др.).

Выполнение арифметических операций сложения, вычитания, умножения, деления, а также побитовые операции могут быть реализованы аппаратным способом. Другие арифметические операции, такие как возведение в степень, взятие корня и др., реализуются программным способом.

2.1 Арифметика целых положительных чисел

Как было отмечено ранее, арифметические операции в двоичной системе счисления с положительными числами выполняются по тем же правилам, что и в десятичной системе счисления. Поскольку в двоичной системе счисления используются всего две цифры – нуль и единица, то нагляднее всего можно отобразить эти правила в виде таблицы, где будет всего четыре возможные комбинации результата выполнения операций сложения, вычитания и умножения.

Сложение положительных двоичных чисел

В общем виде правила сложения двоичных положительных чисел выглядят следующим образом:

– два двоичных положительных числа складываются в столбик поразрядно, начиная с младшего разряда;

– при сложении нулевого разряда с нулевым получается сумма, равная нулю (также нулевой разряд);

– при сложении нулевого разряда с единичным получается сумма, равная единице (единичный разряд);

– при сложении единичного разряда с единичным получается сумма, равная нулю, с переносом единицы в старший разряд;

– при переносе единицы из младшего разряда в старший сначала складываются значения соответствующих разрядов двух двоичных чисел по вышеописанным правилам, затем к полученной поразрядной сумме прибавляется перенесенная единица (по вышеописанным правилам).

Наглядно данные правила представлены в таблице 2.1.

Таблица 2.1 – Правила сложения двоичных положительных чисел

+		**Первое слагаемое**	
		0	**1**
Второе слагаемое	**0**	0	1
	1	1	0*
** Происходит перенос единицы в старший разряд.*			

Пример

Выполнить сложение двух положительных двоичных чисел: 10110011101_2 и 11101000110_2.

Выполняем сложение поразрядно в столбик, начиная с младшего разряда.

```
  • • •       • • •
    1 0 1 1 0 0 1 1 1 0 1
+   1 1 1 0 1 0 0 0 1 1 0
-------------------------
1 1 0 0 1 1 1 0 0 0 1 1 .
```

При сложении в третьем разряде происходит переполнение, и единица переходит в четвертый разряд, где также вызывает переполнение. Аналогичная ситуация происходит в пятом, шестом, девятом, десятом и одиннадцатом разрядах. Лишняя единица в одиннадцатом разряде переходит в двенадцатый. Таким образом, конечное значение (сумма) будет иметь уже не одиннадцать, а двенадцать разрядов.

В общем случае при формировании значения в текущем разряде результата приходится дважды применять таблицу 2.1: первый раз при сложении соответствующих разрядов операндов, формируя так называемую поразрядную сумму, и второй раз – для сложения разряда сформированной поразрядной суммы и переноса из ближайшего младшего разряда, если таковое произошло.

Вычитание положительных двоичных чисел

В общем виде правила вычитания двоичных положительных чисел выглядят следующим образом:

– два двоичных положительных числа вычитаются в столбик поразрядно, начиная с младшего разряда;

– при вычитании нуля из нулевого разряда получается разность, равная нулю (также нулевой разряд);

– при вычитании нуля из единичного разряда получается разность, равная единице;

– при вычитании единицы из единичного разряда получается разность, равная нулю (также нулевой разряд);

– при вычитании единицы из нулевого разряда получается разность, равная единице, при этом происходит заем единицы из старшего разряда;

– при вычитании единицы из нулевого разряда, из которого уже происходил заем единицы, получается разность, равная нулю, и при этом также происходит заем единицы из старшего разряда;

– при вычитании нуля из нулевого разряда, из которого уже происходил заем единицы, получается разность, равная единице, и при этом также происходит заем единицы из старшего разряда.

Наглядно данные правила представлены в таблице 2.2.

Таблица 2.2 – Правила вычитания двоичных положительных чисел

–		**Уменьшаемое**		
		0	**0, из которого взята единица в младший разряд**	**1**
Вычитаемое	**0**	0	1^*	1
	1	1^*	0^*	0
**** Происходит заем единицы из старшего разряда.***				

Пример

Найти разность двух положительных двоичных чисел 1100101_2 и 101110_2. Выполняем вычитание поразрядно в столбик, начиная с младшего разряда.

$$\begin{array}{cccccccc} & \bullet & \bullet & \bullet & \bullet & \bullet & & \\ & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ - & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ \hline & 0 & 1 & 1 & 0 & 1 & 1 & 1 \end{array} .$$

При вычитании из нуля единицы во втором разряде происходит заем единицы из старшего, третьего, разряда, а разность получается, равная единице. Таким образом, после займа в третьем разряде, для вычитания необходимо провести заем единицы из старшего, четвертого, разряда, а разность получается, равная единице. При вычитании в четвертом разряде происходит заем единицы из старшего, пятого, разряда, и т. к. уменьшаемое в четвертом разряде – это «нулевой» разряд, из которого уже происходил заем в младший разряд, а вычитаемое – единица, то разность будет равна нулю. При вычитании в пятом разряде происходит заем единицы из старшего, шестого, разряда, и т. к. уменьшаемое в пятом разряде – это «нулевой» разряд, из которого уже происходил заем в младший разряд, а вычитаемое – тоже нуль, то разность будет равна единице. При вычитании в шестом разряде происходит заем единицы из старшего седьмого разряда, а разность получается, равная единице. В седьмом разряде уменьшаемым и вычитаемым будут являться нулевые разряды, поэтому разность будет равняться нулю.

Умножение двоичных положительных чисел

В общем виде правила умножения двоичных положительных чисел выглядят следующим образом:

– два двоичных положительных числа умножаются в столбик поразрядно, при этом формирование произведения выполняется за счет суммирования частичных произведений, которые формируются посредством умножения множимого на отдельные разряды множителя с учетом веса соответствующего разряда множителя;

– умножение может выполняться как начиная с младшего разряда, так и начиная со старшего (в отличие от десятичной арифметики);

– частичное произведение для разряда множителя равняется нулю, если этот разряд равен нулю;

– частичное произведение для разряда множителя равняется множимому, взятому с соответствующим весом, если разряд множителя равен единице;

– суммирование частичных произведений производится также в столбик с учетом веса соответствующего разряда и сдвига влево или вправо на один разряд каждого частичного произведения при умножении с младшего или старшего разряда соответственно по вышеописанным правилам сложения.

Наглядно данные правила представлены в таблице 2.3.

Таблица 2.3 – Правила умножения двоичных положительных чисел

×		**Множимое**	
		0	**1**
Множитель	**0**	0	0
	1	0	1

Пример

Найти произведение двух положительных двоичных чисел: 1001011_2 и 110101_2, начиная формирование частичных произведений с младшего разряда.

Выполняем умножение поразрядно в столбик, начиная с младшего разряда.

						×	1	0	0	1	0	1	1	– *множимое*
							1	1	0	1	0	1		– *множитель*
						1	0	0	1	0	1	1		– *1-е частичное произведение*
					+	0	0	0	0	0	0	0		– *2-е частичное произведение*
					0	1	0	0̇	1	0	1	1		– *сумма 1-го и 2-го произведений*
				+	1	0	0	1	0	1	1			– *3-е частичное произведение*
				1	0	1	1	1	0	1	1	1		– *сумма 1-го, 2-го и 3-го произведений*
			+	0	0	0	0	0	0	0				– *4-е частичное произведение*
			0̇	1̇	0̇	1̇	1̇	1	0	1	1	1		– *сумма 1-го, 2-го, 3-го и 4-го произведений*
		+	1	0	0	1	0	1	1					– *5-е частичное произведение*
		1	1	0	1̇	0̇	1	0	0	1	1	1		– *сумма 1-го, 2-го, 3-го, 4-го и 5-го произведений*
+	1	0	0	1	0	1	1							– *6-е частичное произведение*
	1	1	1	1	1	0	0	0	0	1	1	1		– *конечное произведение (сумма всех частичных произведений)*

Таким образом, количество частичных произведений соответствует числу разрядов множителя. В примере точками сверху отмечены разряды, в которые происходит перенос единицы по рассмотренным выше правилам сложения положительных двоичных чисел. Начиная со второго частичного произведения, происходит сдвиг следующего частичного произведения на один разряд влево. Количество разрядов частичных произведений соответствует количеству разрядов множимого.

Пример

Найти произведение двух положительных двоичных чисел: 1001011_2 и 110101_2, начиная формирование частичных произведений со старшего разряда.

Выполняем умножение поразрядно в столбик, начиная со старшего разряда.

×	1	0	0	1	0	1	1						*– множимое*
		1	1	0	1	0	1						*– множитель*
+	1	0	$\dot{0}$	$\dot{1}$	$\dot{0}$	$\dot{1}$	1						*– 1-е частичное произведение*
		1	0	0	1	0	1	1					*– 2-е частичное произведение*
+	1	1	1	0	0	0	0	1					*– сумма 1-го и 2-го произведений*
			0	0	0	0	0	0	0				*– 3-е частичное произведение*
+	1	1	1	0	0	0	0	1	0				*– сумма 1-го, 2-го и 3-го произведений*
				1	0	0	1	0	1	1			*– 4-е частичное произведение*
+	1	1	1	1	0	0	1	1	1	1			*– сумма 1-го, 2-го, 3-го и 4-го произведений*
					0	0	0	0	0	0	0		*– 5-е частичное произведение*
+	1	1	1	1	$\dot{0}$	$\dot{0}$	$\dot{1}$	$\dot{1}$	1	1	0		*– сумма 1-го, 2-го, 3-го, 4-го и 5-го произведений*
						1	0	0	1	0	1	1	*– 6-е частичное произведение*
	1	1	1	1	1	0	0	0	0	1	1	1	*– конечное произведение (сумма всех частичных произведений)*

Аналогично количество частичных произведений соответствует числу разрядов множителя. Начиная со второго частичного произведения, происходит сдвиг следующего частичного произведения на один разряд вправо. Количество разрядов частичных произведений соответствует количеству разрядов множимого. Следует отметить, что при суммировании частичных произведений, сложение производится по вышеописанным правилам, т. е. с младшего разряда, и при переполнении единица переносится в старший разряд. В примере точками сверху отмечены разряды, в которые происходит перенос единицы.

Как можно заметить, вне зависимости от того, с младшего или со старшего разряда производится умножение, результат получается одинаковый.

Умножение также можно выполнять без перемножения нулевых разрядов множителя, поскольку они не влияют на конечную сумму, но при этом учитывая соответствующий вес этих разрядов (дополнительный сдвиг влево при умножении с младшего разряда или вправо при умножении со старшего разряда). Также можно не выполнять промежуточное суммирование, а сложить все полученные частичные произведения в конце, учитывая, что каждая пара единичных разрядов в сумме дает нуль и один перенос в старший разряд.

Пример умножения без перемножения нулевых разрядов множителя и промежуточного суммирования представлен ниже.

```
                  ×  1 0 0 1 0 1 1     – множимое
                     1 1 0 1 0 1       – множитель
                  ---------------------
                     Ï Ò Ò 1 0 1 1     – 1-е частичное произведение
               +  İ Ö 0 1 0 1 1 *      – 3-е частичное произведение
            +  1 Ò 0 1 0 1 1 *         – 5-е частичное произведение
         + 1 0 0 1 0 1 1               – 6-е частичное произведение
         ---------------------------
         1 1 1 1 1 0 0 0 0 1 1 1       – конечное произведение (сумма всех ненулевых
                                         частичных произведений)
```

В рассмотренном примере символом «звездочка» (*) отмечены строки с частичными произведениями, где пропускалось умножение на нулевой разряд множителя и делался дополнительный сдвиг влево за каждый пропущенный разряд.

При этом следует отметить, что в случае если младшие разряды множителя являются нулевыми, то в качестве соответствующих младших разрядов произведения следует записать нуль, что показано в примере ниже стрелками.

```
                  ×  1 0 0 1 0 1 1     – множимое
                     1 0 0 1 0 0       – множитель
                  ---------------------
                                  0    – 1-е частичное произведение
                  +            0 |     – 2-е частичное произведение
               +  1 Ò Ò 1 0 1 1 |       – 3-е частичное произведение
            +  1 0 Ò 1 0 1 1 * * |      – 6-е частичное произведение
            ---------------------------
            1 0 1 0 1 0 0 0 1 1 0 0     – конечное произведение (сумма
                                         всех частичных произведений)
```

При умножении правильных двоичных дробей действуют те же правила. При умножении одного n-разрядного числа на другое n-разрядное число получается $(2 \cdot n)$-разрядное произведение, в котором младшие n разрядов можно округлять. Запятая, отделяющая целую и дробные части числа, игнорируется при формировании частичных произведений и ставится только в самом конце, когда получено конечное произведение.

Пример

Найти произведение двух положительных правильных двоичных дробей: $0{,}110_2$ и $0{,}011_2$.

Выполняем умножение поразрядно в столбик, начиная с младшего разряда.

```
                  ×  0, 1 1 0     – множимое
                     0, 0 1 1     – множитель
                  ---------------
                     0 1 1 0      – 1-е частичное произведение
               +  Ò İ 1 0         – 2-е частичное произведение
            ---------------------
               1 0 0 1 0          – сумма 1-го и 2-го произведений
            +  0 0 0 0            – 3-е частичное произведение
            ---------------------
               0 1 0 0 1 0        – сумма 1-го, 2-го и 3-го произведений
         +  0 0 0 0               – 4-е частичное произведение
         ---------------------------
         0, 0 1 0 0 1 0           – конечное произведение
```

Аналогичным образом можно умножать правильные двоичные положительные дроби, начиная со старшего разряда.

Деление двоичных положительных чисел

Деление в любой системе счисления в общем случаев является неточной операцией, поэтому при ее выполнении прежде всего устанавливается количество разрядов частного, которые подлежат определению.

Деление в двоичной системе счисления может выполняться точно так же, как и в десятичной, однако формирование частного двоичных операндов реализуется гораздо проще, т. к.:

– упрощается процедура подбора очередной цифры (в двоичной системе очередной цифрой может быть одна из двух: либо нуль, либо единица);

– упрощается процедура умножения найденной цифры частного на делитель.

В общем виде правила деления двоичных положительных чисел выглядят следующим образом:

1 Два двоичных положительных числа делятся в столбик, как и десятичные. Сначала в делимом от старшего разряда берется n разрядов, где n – число разрядов в делителе.

2 Определяется количественная оценка числа, которое образуют эти старшие n разрядов делимого.

2.1 Если это число больше делителя, то записывается единичный старший разряд частного, а от этих n старших разрядов делимого отнимается делитель, получается первая частичная разность.

2.2 Если это число меньше делителя, то записывается нулевой старший разряд частного, а от этих n старших разрядов делимого отнимается n нулевых разрядов, получается первая частичная разность.

3 Далее к полученным частичным разностям добавляется (спускается) следующий старший разряд делимого.

3.1 Если после добавления последующего старшего разряда делимого к частичной разности получается число больше делителя, то записывается единичный старший разряд частного и вычитается очередная частичная разность.

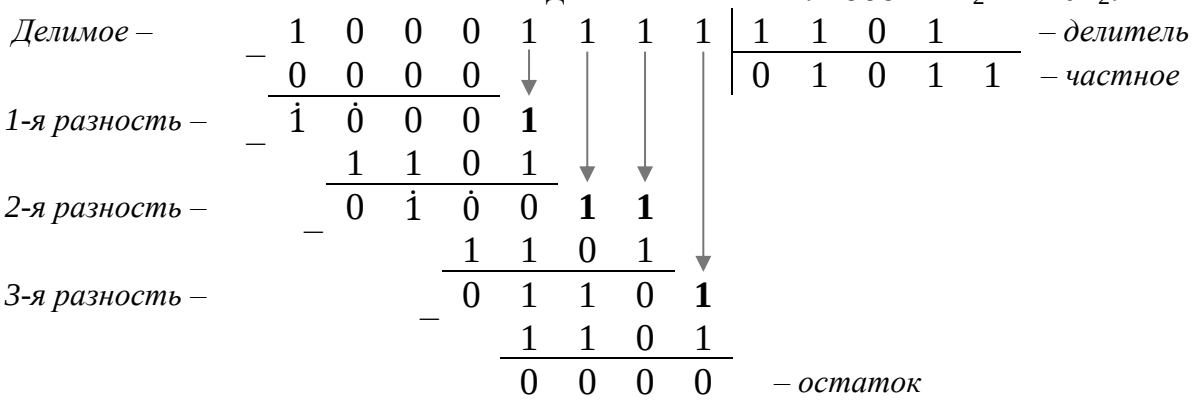
3.2 Если после добавления последующего старшего разряда делимого к частичной разности получается число меньше делителя, то добавляется (спускается) следующий старший разряд делимого, а в частное записывается нулевой разряд и вычитается очередная частичная разность.

4 Вычитание частичных разностей выполняется по правилам вычитания положительных двоичных чисел, описанным выше.

5 Процесс деления продолжается, пока к очередной частичной разности не будет добавлен (спущен) самый младший разряд делимого и не будет получен остаток от деления (нулевой, если делимое делится на делитель без остатка, или ненулевой – в противном случае).

Пример

Найти частное положительных двоичных чисел: 10001111_2 и 1101_2.



В примере точками сверху отмечены старшие разряды, в которых происходит заем единицы по рассмотренным выше правилам вычитания положительных двоичных чисел. Стрелками и полужирным шрифтом отмечены очередные добавляемые (спускаемые) к частичной разности разряды делимого.

2.2 Арифметика двоично-десятичных положительных чисел

Сложение и вычитание положительных двоично-десятичных чисел

В отличие от остальных арифметик позиционных систем счисления двоично-десятичная арифметика имеет свои особенности, выраженные в необходимости корректировки результата арифметических операций путем добавления в тетраду двоичной шестерки (в двоичном коде – 0110) при переносе в старший разряд или при займе из старшего разряда, а также при превышении в тетраде значения, большего девяти.

Двоично-десятичная система счисления представляет собой синтез двоичной и десятичной систем счисления. Каждый разряд десятичного числа (каждая цифра) представляется в виде двоичной тетрады (четырех разрядов). Арифметические операции сложения и вычитания в двоично-десятичной системе счисления выполняются по правилам двоичной арифметики, рассмотренным выше, с последующими определенными корректировками.

Правила коррекции при сложении и вычитании двоично-десятичных чисел в общем виде выглядят следующим образом:

– необходимо добавить двоичную шестерку (0110) в те тетрады, из которых был перенос при их переполнении при двоичном сложении. Необходимость такой коррекции обусловливается тем, что по правилам двоичного суммирования при переносе из младшей тетрады в старшую переносится значение, равное шестнадцати, а по правилам десятичного суммирования должно переноситься значение, равное десяти. То есть перенос, сформированный по правилам двоичной арифметики, убрал из младшей тетрады значение, на шестерку большее, чем нужно;

– необходимо вычесть двоичную шестерку (0110) из тех тетрад, из которых произошел заем в младшие тетрады. Это обусловливается тем, что заем, сформи-

рованный по правилам двоичного вычитания, приносит в тетраду значение, равное шестнадцати, а для десятичного вычитания заем должен был принести в тетраду значение, равное десяти. То есть заем, сформированный по правилам двоичного вычитания, принес значение на шестерку больше нужного;

– необходимо добавить двоичную шестерку (0110) в те тетрады, в которых получено значение больше девяти. Такая коррекция обусловливается тем, что по правилам десятичной арифметики в таких тетрадах должен быть выработан перенос, и чтобы его выработать по правилам двоичной арифметики, в тетраду нужно добавить значение, равное шести.

Пример

Найти сумму десятичных чисел $A = 3927_{10}$ и $B = 4856_{10}$, а затем найти разность этих же чисел B и A, используя двоично-десятичную систему счисления.

В соответствие каждому разряду десятичных чисел A и B находим двоичные тетрады:

$A_{10} =$ 3 9 2 7; $B_{10} =$ 4 8 5 6.

$A_{2–10} =$ 0011 1001 0010 0111; $B_{2–10} =$ 0100 1000 0101 0110.

Складываем двоично-десятичные числа по правилам двоичной арифметики и в необходимых тетрадах выполняем корректировку:

		▲		■	
+	0̇0̇1̇1̇	1001	0̇0̇1̇0̇	0̇1̇11	*– число $A_{2–10}$*
	0100	1000	0101	0110	*– число $B_{2–10}$*
	1000	0001	0̇1̇1̇1̇	1̇101	*– двоичная сумма*
+		0110		0110	*– коррекция*
	1000	0111	1000	0011	*– двоично-десятичная сумма 3927 + 4856 = 8783*

В примере треугольником обозначена тетрада, в которой произошло переполнение и перенос в старшую тетраду, а квадратом обозначена тетрада, в которой в результате двоичного сложения получилось значение больше девяти (тетрада превысила значение 1001). По правилам в этих тетрадах необходимо провести коррекцию (прибавить к ним двоичную шестерку). Точками обозначены разряды, в которые сделан перенос из младших разрядов по правилам двоичной арифметики.

Вычитаем двоично-десятичные числа B и A по правилам двоичной арифметики и в необходимых тетрадах выполняем корректировку:

		♦		♦	
–	0̇1̇0̇0̇	1̇0̇0̇0	0̇1̇0̇1̇	0̇1̇1̇0	*– число $B_{2–10}$*
	0011	1001	0010	0111	*– число $A_{2–10}$*
	0000	1111	0010	1111	*– двоичная разность*
–		0110		0110	*– коррекция*
	0000	1001	0010	1001	*– двоично-десятичная разность 4856 – 3927 = 0929*

В примере ромбами обозначены тетрады, в которые взят заем из старших тетрад. По правилам в этих тетрадах необходимо провести коррекцию (вычесть из них двоичную шестерку). Точками обозначены разряды, из которых взят заем в младшие разряды по правилам двоичной арифметики.

2.3 Поразрядные (побитовые) операции

Поразрядные (побитовые) операции – это операции, которые выполняются над разрядами операндов с одинаковым номером разряда, невзирая на соседние разряды.

К поразрядным операциям относятся:

– поразрядное дополнение;
– поразрядное сложение по модулю 2;
– поразрядное логическое сложение;
– поразрядное логическое умножение;
– логический сдвиг (влево/вправо);
– циклический сдвиг (влево/вправо);
– арифметический сдвиг (влево/вправо);
– операция нормализации.

Логические операции

Операции сдвига

Первые четыре операции относятся к логическим и реализуют одну из логических функций (отрицание, сложение, исключающее сложение, умножение). Последние четыре операции относятся к операциям сдвига и заключаются в одновременном смещении разрядов числа влево или вправо на фиксированное значение по определенному правилу.

Поразрядное дополнение (логическое отрицание (*NOT*))

Поразрядное дополнение реализует логическую операцию отрицания и по сути представляет собой операцию инверсии, т. е. единичные разряды заменяются на нулевые, а нулевые – на единичные.

Пример поразрядного дополнения представлен на рисунке 2.1.

До операции поразрядного дополнения

1	0	1	1	0	1	0	1

После операции поразрядного дополнения

0	1	0	0	1	0	1	0

Рисунок 2.1 – Пример поразрядного дополнения

Операция поразрядного дополнения используется, например, для получения обратного кода числа из прямого кода числа или наоборот.

Поразрядное сложение по модулю 2 (логическая операция исключающее ИЛИ (*XOR*))

Реализует логическую операцию исключающего ИЛИ и используется для проверки двух чисел на равенство и представляет собой сложение двух разрядов по следующим правилам:

1) $0 \oplus 0 = 0$;
2) $0 \oplus 1 = 1$;

3) $1 \oplus 0 = 1$;
4) $1 \oplus 1 = 0$.

Поразрядное логическое сложение (логическая операция ИЛИ (*OR*))

Реализует логическую операцию ИЛИ, т. е. результат выполнения логического сложения двух разрядов будет равен единице, если хотя бы один из них равен единице (или первый, или второй, или оба одновременно):

1) $0 + 0 = 0$;
2) $0 + 1 = 1$;
3) $1 + 0 = 1$;
4) $1 + 1 = 1$.

Поразрядное логическое умножение (логическая операция И (*AND*))

Реализует логическую операцию И, т. е. результат выполнения логического умножения двух разрядов будет равен единице, если оба из них равны единице (и первый, и второй):

1) $0 \cdot 0 = 0$;
2) $0 \cdot 1 = 0$;
3) $1 \cdot 0 = 0$;
4) $1 \cdot 1 = 1$.

Логический сдвиг (влево/вправо)

Логический сдвиг влево заключается в одновременном смещении влево на *n* разрядов всех цифр числа (включая знак), при этом освободившиеся справа *n* разрядов заполняются нулями, а вышедшие за пределы разрядной сетки слева *n* разрядов «исчезают».

Логический сдвиг вправо заключается в одновременном смещении вправо на *n* разрядов всех цифр числа (включая знак), при этом освободившиеся слева *n* разрядов заполняются нулями, а вышедшие за пределы разрядной сетки справа *n* разрядов «исчезают».

Примеры логического сдвига представлены на рисунке 2.2.

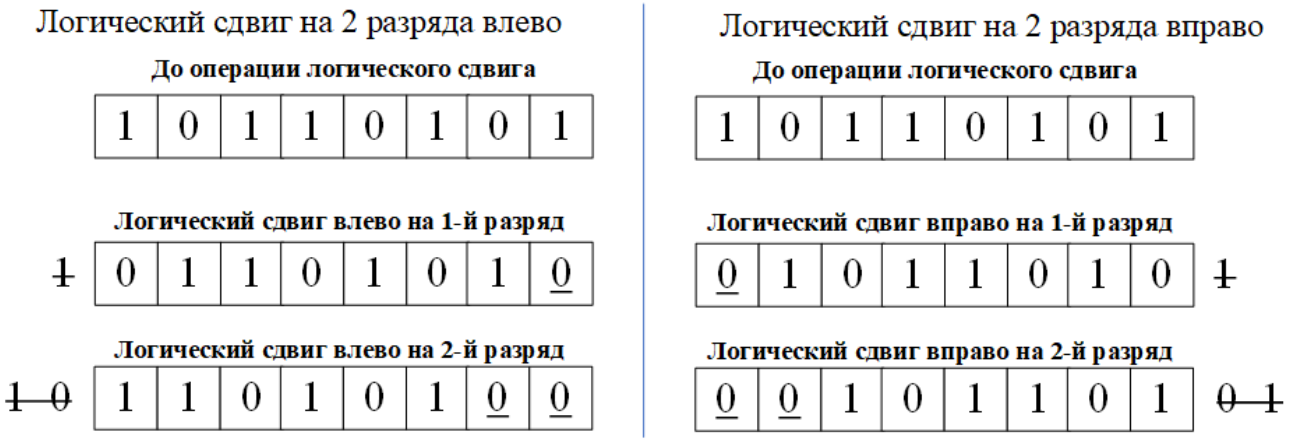


Рисунок 2.2 – Примеры логических сдвигов на два разряда

Циклический сдвиг (влево/вправо)

Циклический сдвиг влево заключается в одновременном смещении влево на *n* разрядов всех цифр числа (включая знак), при этом освободившиеся справа *n* разрядов заполняются сдвинутыми значениями разрядов слева.

Циклический сдвиг вправо заключается в одновременном смещении вправо на *n* разрядов всех цифр числа (включая знак), при этом освободившиеся слева *n* разрядов заполняются сдвинутыми значениями разрядов справа.

Таким образом, сдвигаемые биты перемещаются на место освобождающихся (вне зависимости от направления сдвига). Примеры циклического сдвига представлены на рисунке 2.3.

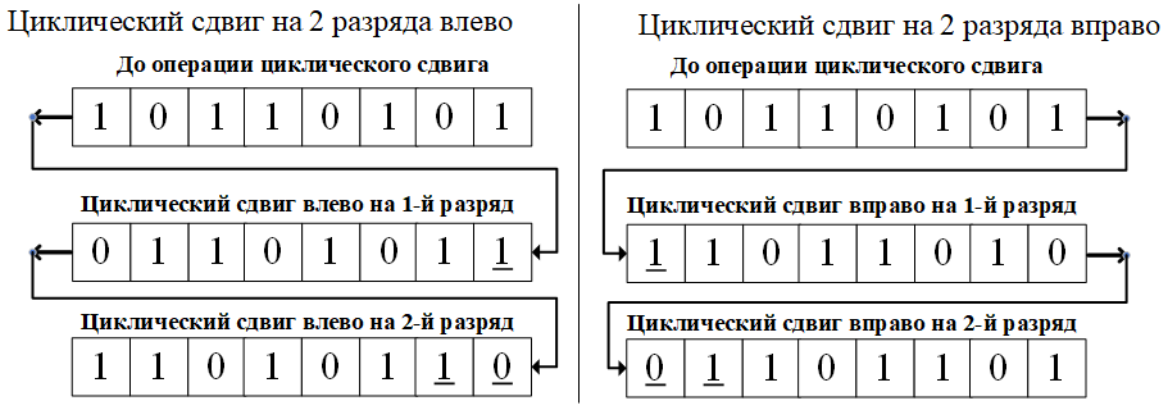


Рисунок 2.3 – Примеры циклических сдвигов на два разряда

Арифметический сдвиг (влево/вправо)

При арифметическом сдвиге влево беззнакового (положительного) числа освобождающиеся справа разряды заполняются нулями, а выдвигаемые слева не исчезают и становятся старшими разрядами (увеличивается разрядность числа на величину сдвига).

При арифметическом сдвиге вправо беззнакового (положительного) числа выдвигаемые справа разряды «исчезают», а освобождающиеся слева разряды заполняются нулями (уменьшается разрядность числа на величину сдвига, поскольку старшие (правые) нулевые разряды не значащие).

Примеры арифметического сдвига беззнаковых чисел представлены на рисунке 2.4.

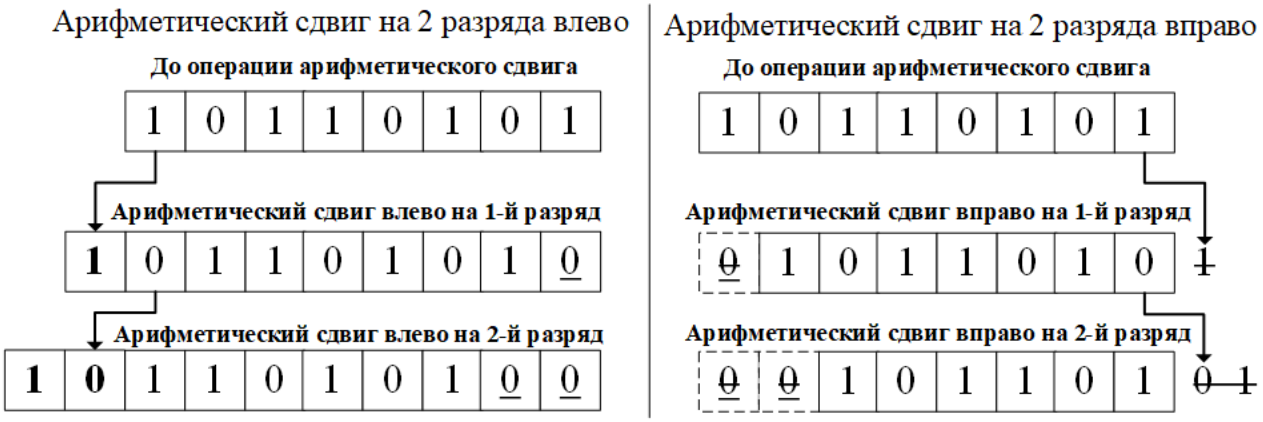


Рисунок 2.4 – Примеры арифметических сдвигов на два разряда беззнаковых чисел

Если число знаковое, то арифметический сдвиг осуществляется без изменения позиции знакового бита, т. е. сдвигается только модульное поле.

При арифметическом сдвиге знакового числа влево освобождающиеся правые разряды модульного поля заполняются нулями, а сдвигаемые левые разряды «исчезают».

При арифметическом сдвиге знакового числа вправо освобождающиеся слева разряды заполняются значением знакового (самого левого) бита, а выдвигаемые правые разряды «исчезают». Примеры арифметического сдвига знаковых чисел представлены на рисунке 2.5.

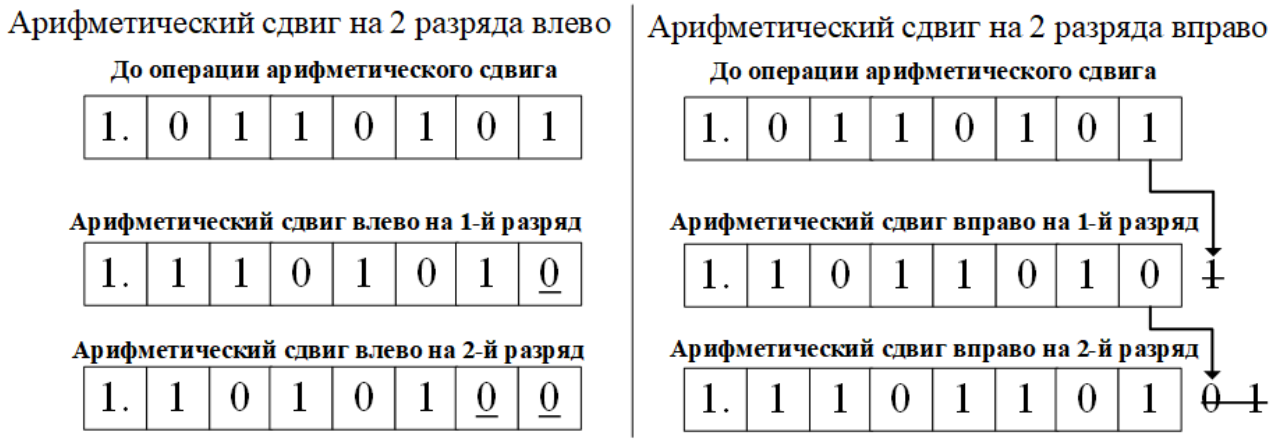


Рисунок 2.5 – Примеры арифметических сдвигов на два разряда знаковых чисел

Арифметический сдвиг знакового числа эквивалентен умножению числа на основание системы счисления, возведенное в степень (положительную при сдвиге влево или отрицательную при сдвиге вправо), равную величине сдвига (количеству разрядов, на которое выполняется сдвиг). Например, для знакового однобайтового числа 0.1011010_2:

– арифметический сдвиг влево на 2 разряда: $0.1011010 \cdot 2^2 = 0.1101000$;

– арифметический сдвиг вправо на 3 разряда: $0.1011010 \cdot 2^{-3} = 0.0001011$.

Операция нормализации

Поразрядная операция нормализации выполняется над вещественными числами с плавающей точкой, если число оказалось ненормализованным в процессе вычислений.

Операция нормализации влево (значит, произошло нарушение нормализации справа от точки) заключается в выполнении арифметического сдвига мантиссы влево поразрядно, пока число не будет нормализовано. При этом каждый сдвиг мантиссы на один разряд влево будет сопровождаться вычитанием единицы из показателя степени.

Операция нормализации вправо (значит, произошло нарушение нормализации слева от точки) заключается в выполнении арифметического сдвига мантиссы вправо поразрядно, пока число не будет нормализовано. При этом каждый сдвиг мантиссы на один разряд вправо будет сопровождаться увеличением на единицу показателя степени.

В языках программирования высокого уровня (*Delphi*, *C*/*C*++) поразрядные операции определены только над целочисленными операндами. Примеры записи и применения поразрядных операций в синтаксисе языков программирования *Delphi* и *C*/*C*++ представлены в таблице 2.4.

Таблица 2.4 – Примеры поразрядных операций в языках программирования

Оператор	В синтаксисе *Delphi*	В синтаксисе *C*/*C*++	Пример выполнения
Поразрядное дополнение целого	`not 12;`	`~ 12;`	1-байтовый беззнаковый тип в ПК[1]: $12_{10} = 00001100_2$ $11110011_2 = 243_{10}$ (в ПК[1]) 1-байтовый знаковый тип в ДК[2]: $12_{10} = 00001100_2$ $11110011_2 = -13_{10}$ (в ДК[2])
Поразрядное логическое умножение (И)	`12 and 5;`	`12 & 5;`	1-байтовый беззнаковый тип в ПК[1]: $12_{10} = 00001100_2$ $5_{10} = 00000101_2$ $00000100_2 = 4_{10}$ (в ПК[1])
Поразрядное логическое сложение (ИЛИ)	`12 or 5;`	`12 | 5;`	1-байтовый беззнаковый тип в ПК[1]: $12_{10} = 00001100_2$ $5_{10} = 00000101_2$ $00001101_2 = 13_{10}$ (в ПК[1])
Поразрядное логическое исключающее ИЛИ	`a xor b;`	`a ^ b;`	1-байтовый беззнаковый тип в ПК[1]: $12_{10} = 00001100_2$ $5_{10} = 00000101_2$ $00001001_2 = 9_{10}$ (в ПК[1])

Примечания
[1] ПК – прямой код представления целых чисел.
[2] ДК – дополнительный код представления целых чисел.

2.4 Арифметика целых двоичных алгебраических чисел

Все арифметические операции над двоичными алгебраическими числами выполняются *только в обратном или дополнительном коде*.

При использовании обратного или дополнительного кода операция *вычитания заменяется на операцию сложения с изменением знака второго операнда* (изменением значения разряда знакового поля на противоположное) и сложение осуществляется по правилам двоичной арифметики (см. подраздел 2.1).

Сложение алгебраических чисел в обратном и дополнительном кодах

Поскольку вычитание в обратном и дополнительном коде заменяется на операцию сложения с изменением знака вычитаемого числа, то правила сложения в обратном и дополнительном кодах выглядят следующим образом:

1 При сложении чисел, представленных в обратном или дополнительном коде, выполняется сложение разрядов, представляющих запись операндов, по правилам двоичной арифметики (в столбик поразрядно, начиная с младшего разряда) по всей длине записи чисел, не обращая внимание на границу, разделяющую знаковое и модульные поля (точку).

2 В результате такого сложения может возникнуть переполнение знакового и модульного полей. Для определения возникновения любого из видов переполнения необходимо обратить внимание на знаковое поле.

3 В первую очередь обрабатывается переполнение знакового поля. О переполнении знакового поля свидетельствует большее количество разрядов в знаковом поле, чем положено (более одного разряда в базовых кодах и более двух – в модифицированных). Переполнение разрядности знакового поля, т. е. перенос, возникший из крайнего разряда модульного поля, обрабатывается в разных кодах по-разному.

3.1 В обратном (базовом или модифицированном) коде переполнение (старший разряд) знакового поля учитывается как плюс единица в младший разряд модульного поля полученной суммы.

3.2 В дополнительном (базовом или модифицированном) коде переполнение (старший разряд) знакового поля игнорируется (никак не учитывается, просто исчезает).

4 Переполнение модульного поля можно идентифицировать только в модифицированных кодах. После обработки переполнения знакового поля (если оно имело место быть) необходимо снова обратить внимание на знаковое поле. Если в нем в модифицированных кодах будут различные разряды (01 или 10) – это свидетельствует о переполнении модульного поля. Для обработки переполнения модульного поля необходимо младший разряд знакового поля перенести в качестве старшего разряда модульного поля, т. е. за границу (точку), разделяющую знаковое и модульное поля. Таким образом разрядность модульного поля увеличится на единицу. Оставшийся же старший разряд знакового поля является знакоопределяющим.

Суть переполнения модульного поля можно пояснить на примере десятичной арифметики (для удобства понимания), записав слагаемые как бы в модифицированном коде (знаки операндов двумя разрядами). Например, сложим два положительных числа одинаковой разрядности: ++99 и ++11.

		Поле знака		*Поле модуля*
Операция сложения →	+	+ +	.	9 9
		+ +	.	1 1
		+ 1	.	1 0

Разряд знака ↗ ↖ *Разряд модуля числа*

После обработки переполнения модульного поля:

	+ +	.	1 1 0 ← *Увеличение разрядности модульного поля*
	Поле знака		*Поле модуля*

Данный пример иллюстрирует, что при сложении разряд модуля числа может оказаться в знаковом поле. В двоичной же арифметике и знаки числа, и разряды модуля числа записываются нулями и единицами, поэтому становится трудно понять, что произошло переполнение.

Если же не использовать модифицированный код, то можно заранее увеличить разрядность модульного поля на один старший нулевой разряд. Это позволит предупредить переполнение модульного поля, и при этом разряд модуля не попадет в знаковое поле.

Пример

Сформировать запись десятичных чисел $A = 182_{10}$ и $B = -107_{10}$ в прямом, обратном и дополнительном базовых двоичных кодах и найти значения чисел $C_1 = A + B$, $C_2 = A - B$, используя обратный код, а также $C_3 = B - A$, $C_4 = -A - B$, используя дополнительный код. Все результаты арифметических операций представить в прямом коде.

Сначала переводим десятичные числа A и B в двоичные:

$A = 182_{10} = +10110110_2$, $B = -107_{10} = -1101011_2$.

Число A имеет восемь разрядов, а число B – семь разрядов. Поскольку над этими числами будут выполняться арифметические операции, необходимо привести их к одинаковому количеству разрядов модульной части, учитывая также ожидаемый результат выполнения заданных арифметических операций. Таким образом, количество разрядов модульной части представления чисел в различных кодах должно быть равным девяти. То есть число A дополняется одним старшим нулевым разрядом, а число B дополняется двумя старшими нулевыми разрядами:

$A = 182_{10} = +010110110_2$, $B = -107_{10} = -001101011_2$.

Далее формируем запись чисел в прямом, обратном и дополнительном кодах согласно (1.23), (1.24) и (1.25). Поскольку число A положительное, его представление в обратном и дополнительном кодах будет совпадать с представлением в прямом коде:

$[A]_{ПК} = [A]_{ОК} = [A]_{ДК} = 0.010110110$.

Число B отрицательное и будет иметь следующее представление в прямом коде:

$[B]_{ПК} = 1.001101011$.

Для определения модульной части числа B в обратном коде прибавим к включенной границе диапазона ($2^n - 1 = 111111111$) число B (по формуле (1.24)):

111111111 + (–001101011) = 110010100.

Таким образом, $[B]_{ОК} = 1.110010100$.

Для определения модульной части числа B в дополнительном коде прибавим к невключенной границе диапазона ($2^n = 1000000000$) число B (по формуле (1.25)):

1000000000 + (–001101011) = 110010101

Таким образом, $[B]_{ДК} = 1.110010101$.

Поскольку в обратном и дополнительном кодах операция вычитания заменяется на операцию сложения с заменой знака вычитаемого на противоположный, приведем необходимые операции в соответствующий вид:

$C_1 = A + B$, $C_2 = A + (-B)$, $C_3 = B + (-A)$, $C_4 = (-A) + (-B)$.

Находим необходимые значения чисел:

$[A]_{ПК} = 0.010110110$; $[A]_{ОК} = 0.010110110$; $[A]_{ДК} = 0.010110110$;
$[-A]_{ПК} = 1.010110110$; $[-A]_{ОК} = 1.101001001$; $[-A]_{ДК} = 1.101001010$;
$[B]_{ПК} = 1.001101011$; $[B]_{ОК} = 1.110010100$; $[B]_{ДК} = 1.110010101$;
$[-B]_{ПК} = 0.001101011$; $[-B]_{ОК} = 0.001101011$; $[-B]_{ДК} = 0.001101011$.

Находим значение $C_1 = A + B$, используя обратный код:

$[A]_{ОК}$ = 0. 0 1 0 1 1 0 1 1 0
$[B]_{ОК}$ + 1. 1 1 0 0 1 0 1 0 0

Корректировка единицей в младший разряд при переполнении знакового поля: 1 0. 0 0 1 0 0 1 0 1 0 + 1

$[C_1]_{ОК} = [C_1]_{ПК}$ = 0. 0 0 1 0 0 1 0 1 1

В рассмотренном примере нахождения значения числа C_1 возникает переполнение разрядности знакового поля, которое учитывается как плюс единица в младший разряд согласно правилам сложения в обратном коде.

Находим значение $C_2 = A + (-B)$, используя обратный код:

$[A]_{ОК}$ = 0. 0 1 0 1 1 0 1 1 0
$[-B]_{ОК}$ + 0. 0 0 1 1 0 1 0 1 1
$[C_2]_{ОК} = [C_2]_{ПК}$ = 0. 1 0 0 1 0 0 0 0 1

В рассмотренном примере нахождения значения числа C_2 вычитание заменяется на сложение, второе слагаемое инвертируется и берется с противоположным знаком $(-B)$.

Находим значение $C_3 = B + (-A)$, используя дополнительный код:

$[B]_{ДК}$ = 1. 1 1 0 0 1 0 1 0 1
$[-A]_{ДК}$ + 1. 1 0 1 0 0 1 0 1 0
$[C_3]_{ДК}$ = 1 1. 0 1 1 0 1 1 1 1 1 – *переполнение знакового поля игнорируется*
$[C_3]_{ПК}$ = 1. 1 0 0 1 0 0 0 0 1

В рассмотренном примере нахождения значения числа C_3 вычитание заменяется на сложение, второе слагаемое инвертируется и берется с противоположным знаком $(-A)$. Переполнение разрядности знакового поля по правилам сложения в дополнительном коде игнорируется. В результате получается отрицательная сумма в дополнительном коде, которую необходимо проинвертировать и добавить единицу в младший разряд для перевода в прямой код.

Находим значение $C_4 = (-A) + (-B)$, используя дополнительный код:

$[-A]_{ДК}$ = 1. 1 0 1 0 0 1 0 1 0
$[-B]_{ДК}$ + 0. 0 0 1 1 0 1 0 1 1
$[C_4]_{ДК}$ = 1. 1 1 0 1 1 0 1 0 1
$[C_4]_{ПК}$ = 1. 0 0 1 0 0 1 0 1 1

В рассмотренном примере нахождения значения числа C_4 вычитание заменяется на сложение, оба слагаемых инвертируются и берутся с противоположным знаком ($-A$ и $-B$). В результате получается отрицательная сумма в дополнительном коде, которую необходимо проинвертировать и добавить единицу в младший разряд для перевода в прямой код.

Модифицированные коды

Иногда бывает трудно заранее определить разрядность модульной части алгебраических двоичных чисел, особенно при выполнении последовательных арифметических операций. Может возникнуть ситуация, когда при сложении двух чисел с одинаковым знаком в результате переполнения модульного поля получается сумма со знаком, противоположным слагаемым. Выполнение таких операций в модифицированных кодах решает данную проблему.

Если в результате сложения чисел в модифицированном коде полученный результат имеет в поле знака одинаковые значения в обоих разрядах (00 или 11), то переполнения модульного поля нет, если же разряды знакового поля имеют не одинаковые значения (10 или 01), то произошло переполнение модульного поля. При этом, если в поле знака имеет место значение 01, то результат положительный, а если 10, то полученный результат отрицательный (основным носителем знака числа является левый разряд знакового поля).

Пример

Найти значения чисел $C_1 = A + B$, $C_2 = A - B$, используя модифицированный обратный код, а также $C_3 = B - A$, $C_4 = -A - B$, используя модифицированный дополнительный код, если $A = 20_{10} = 10100_2$ и $B = -14_{10} = -1110_2 = -01110_2$.

Поскольку в обратном и дополнительном кодах операция вычитания заменяется на операцию сложения с заменой знака вычитаемого на противоположный, приведем необходимые операции в соответствующий вид:

$C_1 = A + B$, $C_2 = A + (-B)$, $C_3 = B + (-A)$, $C_4 = (-A) + (-B)$.

Находим необходимые значения чисел в модифицированных кодах:

$[A]_{МПК} = 00.10100;$	$[A]_{МОК} = 00.10100;$	$[A]_{МДК} = 00.10100;$
$[-A]_{МПК} = 11.10100;$	$[-A]_{МОК} = 11.01011;$	$[-A]_{МДК} = 11.01100;$
$[B]_{МПК} = 11.01110;$	$[B]_{МОК} = 11.10001;$	$[B]_{МДК} = 11.10010;$
$[-B]_{МПК} = 00.01110;$	$[-B]_{МОК} = 00.01110;$	$[-B]_{МДК} = 00.01110.$

Находим значение $C_1 = A + B$, применяя модифицированный обратный код:

$$
\begin{array}{rl}
[A]_{МОК} = & 0\,\dot{0}.\,1\;0\;1\;0\;0 \\
+\;[B]_{МОК} = & 1\,1.\,1\;0\;0\;0\;1 \\
\hline
\textit{Корректировка единицей в младший разряд при переполнении разрядности знакового поля} & 1\;0\,0.\,0\;0\;1\;\dot{0}\;1 \\
+ & \qquad\qquad\qquad\qquad\rightarrow 1 \\
\hline
[C_1]_{МОК} = [C_1]_{МПК} = & 0\,0.\,0\;0\;1\;1\;0
\end{array}
$$

В рассмотренном примере нахождения значения числа C_1 возникает переполнение разрядности знакового поля, которое учитывается как плюс единица в младший разряд согласно правилам сложения в обратном коде. Переполнения модульного поля не произошло, поскольку разряды знакового поля одинаковы.

Находим $C_2 = A + (-B)$, используя модифицированный обратный код:

$$
\begin{array}{rll}
[A]_{МОК} = & 0\,\dot{0}.\,\dot{1}\;\dot{0}\;1\;0\;0 & \\
+\;[-B]_{МОК} = & 0\,0.\,0\;1\;1\;1\;0 & \\
\hline
\textit{Переполнение модульного поля, т. к. знаковые разряды имеют разное значение} & \underline{0}\;\underline{1}.\,0\;0\;0\;1\;0 & \textit{– сдвинуть точку влево} \\
[C_2]_{МОК} = [C_2]_{МПК} = & 0\,0.\,1\;0\;0\;0\;1\;0 &
\end{array}
$$

В рассмотренном примере нахождения значения числа C_2 возникает переполнение модульного поля, поскольку разряды знакового поля различны. Младший разряд знакового поля необходимо перенести в модульное поле. Значение суммы положительно, поскольку носитель знака – старший разряд знакового поля равен нулю.

Находим значение $C_3 = B + (-A)$, используя модифицированный дополнительный код:

$$[B]_{МДК} = \dot{1}\ 1.\ 1\ 0\ 0\ 1\ 0$$
$$+\ [-A]_{МДК} = \underline{1\ 1.\ 0\ 1\ 1\ 0\ 0}$$

Переполнение знакового поля игнорируется; переполнение модульного поля, т. к. знаковые разряды имеют разное значение

$$\dot{1}\ \underline{1}\ \underline{0}.\ 1\ 1\ 1\ 1\ 0$$

– сдвинуть точку влево

$$[C_3]_{МДК} = 1\ 1.\ 0\ 1\ 1\ 1\ 1\ 0$$
$$[C_3]_{МПК} = 1\ 1.\ 1\ 0\ 0\ 0\ 1\ 0$$

– проинвертировать и добавить единицу в младший разряд

В рассмотренном примере нахождения значения числа C_3 возникает переполнение модульного поля, поскольку разряды знакового поля различны. Младший разряд знакового поля необходимо перенести в модульное поле. Значение суммы отрицательно, поскольку носитель знака – старший разряд знакового поля – равен единице, и для получения записи числа в прямом коде необходимо проинвертировать модульное поле и прибавить единицу в младший разряд модульного поля.

Находим значение $C_4 = (-A) + (-B)$, используя модифицированный дополнительный код:

$$[-A]_{МДК} = \dot{1}\ 1.\ \dot{0}\ \dot{1}\ 1\ 0\ 0$$
$$+\ [-B]_{МДК} = \underline{0\ 0.\ 0\ 1\ 1\ 1\ 0}$$
$$[C_4]_{МДК} = 1\ 1.\ 1\ 1\ 0\ 1\ 0$$

– проинвертировать и добавить единицу в младший разряд

$$[C_4]_{МПК} = 1\ 1.\ 0\ 0\ 1\ 1\ 0$$

В рассмотренном примере нахождения значения числа C_4 переполнение модульного поля не возникает, поскольку разряды знакового поля одинаковы. Значение суммы отрицательно, поскольку оба разряда знакового поля равны единице, и для получения записи числа в прямом коде необходимо проинвертировать модульное поле и прибавить единицу в младший разряд.

2.5 Арифметика вещественных чисел, представленных в формате с фиксированной точкой (запятой)

Арифметическая операция сложения (вычитания) над вещественными числами в формате с плавающей точкой выполняется аналогично рассмотренным правилам и алгоритмам сложения для алгебраических чисел (см. подраздел 2.4).

Арифметическая операция умножения над вещественными числами в формате с плавающей точкой выполняется по следующему алгоритму:

1 Формируется знак искомого произведения:

а) знак произведения будет положительный, если знаки сомножителей совпадают («0» и «0», или «1» и «1»);

б) знак произведения будет отрицательный, если знаки сомножителей различные («0» и «1», или «1» и «0»).

2 Выполняется умножение сомножителей по модулю (т. е. положительных чисел) по правилам и алгоритмам умножения для положительных чисел (см. подраздел 2.1).

3 К полученному на шаге 2 произведению применяется знак, полученный на шаге 1.

Арифметическая операция деления над вещественными числами в формате с плавающей точкой выполняется по следующему алгоритму:

1 Формируется знак искомого частного:

а) знак частного будет положительный, если знаки делимого и делителя совпадают («0» и «0», или «1» и «1»);

б) знак частного будет отрицательный, если знаки делимого и делителя различные («0» и «1», или «1» и «0»).

2 Выполняется деление одним из двух способов:

а) с восстановлением остатка;

б) без восстановления остатка.

3 К полученному на шаге 2 частному применяется знак, полученный на шаге 1.

Ввиду того что формат с фиксированной точкой практически не используется в настоящее время, и с учетом того, что алгоритмы деления с восстановлением остатка и без восстановления остатка аналогичны и для вещественных чисел, представленных в формате с плавающей точкой, данный материал подробно будет изложен в подразделе 2.6.

2.6 Арифметика вещественных чисел, представленных в формате с плавающей точкой (запятой)

Арифметические операции для чисел, представленных в формате с плавающей точкой, в общем выполняются по правилам арифметики алгебраических чисел, но с определенными дополнительными операциями, связанными с формой представления чисел.

Сложение (вычитание) чисел с плавающей точкой (запятой)

Операция сложения (вычитания) чисел с плавающей точкой предполагает наличие одинаковых порядков у операндов, подлежащих суммированию (вычитанию). Именно поэтому сложение (вычитание) таких чисел предполагает три этапа реализации операции:

– выравнивание порядков;

– сложение мантисс операндов, имеющих одинаковые порядки;

– определение нарушения нормализации и ее устранение.

Как и в случае с целыми числами, представленными в формате с фиксированной точкой, арифметические операции выполняются в обратном, дополнительном или их модифицированных кодах. Операция вычитания также заменяется на операцию сложения с изменением знака вычитаемого на противоположный.

Пример

Найти разность C чисел A и B, представленных в формате с плавающей точкой, если эти числа представлены в виде порядков $[A_E]_{ПК} = 1.001$ и $[B_E]_{ПК} = 0.001$ соответственно и мантисс $[A_M]_{ПК} = 1.11001$ и $[B_M]_{ПК} = 0.11100$ соответственно. При вычитании использовать модифицированный дополнительный код.

Сначала необходимо выровнять порядки. Для этого из порядка числа A вычитается порядок числа B. Вычитание также заменяется сложением, при этом B_E заменяется на $–B_E$. Находим $[A_E]_{ДК}$ и $[–B_E]_{ДК}$:

$[A_E]_{ДК} = 1.111$ и $[–B_E]_{ДК} = 1.111$.

Находим разность порядков:

$$
\begin{array}{rl}
 & \dot{1}.\,\dot{1}\;\;\dot{1}\;\;1 \quad – [A_E]_{ДК} \\
+ & 1.\,1\;\;1\;\;1 \quad – [–B_E]_{ДК} \\
\hline
 & \dot{1}\,1.\,1\;\;1\;\;0 \quad – \textit{разность порядков в дополнительном коде} \\
 & 1.\,0\;\;1\;\;0 \quad – \textit{разность порядков в прямом коде}
\end{array}
$$

Так как знак разности порядков отрицательный, то в качестве общего порядка, а следовательно, и предварительного значения порядка искомого результата C_E^*, правильнее взять порядок второго числа B_E. Для того чтобы взять в качестве порядка первого числа порядок второго числа, т. е. в нашем случае увеличив его порядок на два, необходимо мантиссу меньшего числа A_M умножить на $2^{–2}$, т. е. выполнить арифметический сдвиг на два разряда вправо:

$$A_M^* = A_M \cdot 2^{-2} = 1.11001 \cdot 2^{-2} = 1.00110.$$

Таким образом, после выравнивания порядков операндов будем иметь следующую форму представления операндов:

$[A_M^*]_{ПК} = 1.00110$, $[B_M^*]_{ПК} = 1.11100$.

Предварительное значение мантиссы определяется как $C_M^* = A_M^* – B_M^*$.

Определять предварительное значение мантиссы будем в модифицированном дополнительном коде, как указано в условии. Для этого находим значения $[A_M^*]_{МДК}$ и $[–B_M^*]_{МДК}$ (второй операнд берется с отрицательным знаком, поскольку операция вычитания в модифицированном дополнительном коде заменяется на операцию сложения):

$[A_M^*]_{МДК} = 11.11010$, $[–B_M^*]_{МДК} = 11.00100$.

Находим предварительное значение C_M^*:

$$
\begin{array}{rl}
 & \dot{1}\;1.\;1\;1\;0\;1\;0 \quad – [A_M]_{МДК} \\
+ & 1\;1.\;0\;0\;1\;0\;0 \quad – [–B_M]_{МДК} \\
\hline
 & \dot{1}\;1\;0.\;1\;1\;1\;1\;0 \quad – [C_M^*]_{МДК} \quad – \textit{переполнение знакового поля игнорируется} \\
 & 1\;0.\;0\;0\;0\;1\;0 \quad – [C_M^*]_{МПК} \quad – \textit{переполнение модульного поля}
\end{array}
$$

Из записи $[C_M^*]_{МПК}$, полученной после вычитания мантисс операндов с выравненными порядками, видно, что нормализация представления результата нарушена (т. к. произошло переполнение модульного поля). Поэтому для данного примера необходимо выполнить этап устранения нарушения нормализации. В данном случае нарушение нормализации слева от точки, т. к. получено $[C_M^*]_{МПК}$ с ненулевой целой частью (неодинаковые разряды в поле знака использованного модифицированного дополнительного кода).

Поэтому необходимо выполнить этап устранения нарушения нормализации. Для того чтобы привести полученную предварительную мантиссу к нормализованной форме, достаточно ее умножить на 2^{-1}, т. е. выполнить ее арифметический сдвиг вправо. В результате будем иметь конечное значение мантиссы

$$C_M = C_M^* \cdot 2^{-1} = 1.00010 \cdot 2^{-1} = 1.10001.$$

Арифметический сдвиг предварительного значения мантиссы C_M^* сопровождается изменением ранее найденного предварительного значения порядка результата C_E^* на плюс единицу. Находим окончательное значение порядка C_E:

$$\begin{array}{rll} & 0\ 0.\ 0\ \dot{0}\ 1 & -[C_E^*]_{МДК} \\ + & 0\ 0.\ 0\ 0\ 1 & -+1 \\ \hline & 0\ 0.\ 0\ 1\ 0 & -[C_E]_{МДК} = [C_E]_{МПК} \end{array}$$

После устранения нарушения нормализации окончательный результат будет иметь вид C: $\{[C_E]_{ПК} = 0.010, [C_M]_{ПК} = 1.10001\}$.

Умножение чисел с плавающей точкой (запятой)

С точки зрения представления чисел в формате с плавающей точкой поиск произведения C двух таких чисел A и B сводится к поиску произведения на основании порядка и мантиссы множимого и множителя. Для двоичных чисел это имеет вид

$$C = A \cdot B = 2^{A_E} \cdot A_M \cdot 2^{B_E} \cdot B_M = 2^{A_E + B_E} \cdot A_M \cdot B_M = 2^{C_E} \cdot C_M. \qquad (2.1)$$

Из формулы (2.1) следует, что порядок произведения определяется как сумма порядков сомножителей, а мантисса произведения – как произведение мантисс сомножителей. Однако, учитывая то, что при умножении мантисс может произойти нарушение нормализации, в результате указанных действий будет найдено предварительное значение порядка и мантиссы искомого произведения, и окончательное значение произведения будет найдено только после устранения нарушения нормализации.

Последовательность действий нахождения произведения двух чисел, представленных в формате с плавающей точкой, имеет следующий общий вид:

– знак произведения определяется как сумма по модулю двух знаковых разрядов мантисс сомножителей (операция *XOR*);

– определяется предварительное значение порядка произведения посредством суммирования порядков сомножителей;

– предварительное значение мантиссы произведения определяется как произведение модулей мантисс сомножителей;

– устраняется нарушение нормализации мантиссы произведения (если нарушение присутствует) соответствующей корректировкой предварительного значения порядка и мантиссы искомого произведения;

– при формировании мантиссы произведения нормализованных чисел с плавающей точкой возможен только один вид нарушения нормализации – нарушение нормализации справа от точки с появлением нуля только в старшем разряде мантиссы.

Пример

Найти произведение C двух чисел A и B, представленных в формате с плавающей точкой, если эти числа представлены в виде порядков $[A_E]_{ПК} = 1.010$ и $[B_E]_{ПК} = 0.001$ соответственно и мантисс $[A_M]_{ПК} = 1.1010$ и $[B_M]_{ПК} = 0.1001$ соответственно. При выполнении операций использовать обратный код. При умножении мантисс использовать метод умножения, начиная с младшего разряда множителя.

Сначала определяем знак искомого произведения. Так как знаки мантисс сомножителей неодинаковые, знак произведения будет отрицательным.

Предварительное значение порядка произведения C_E^* определяется как сумма порядков сомножителей $C_E^* = A_E + B_E$:

$$
\begin{array}{rl}
 & 1.\ 1\ \dot{0}\ 1 \quad -[A_E]_{ОК} \\
+ & 0.\ 0\ 0\ 1 \quad -[B_E]_{ОК} \\
\hline
 & 1.\ 1\ 1\ 0 \quad -[C_E^*]_{ОК} \\
 & 1.\ 0\ 0\ 1 \quad -[C_E^*]_{ПК}
\end{array}
$$

Абсолютное значение предварительного значения мантиссы произведения C_M^* определяется как произведение модулей мантисс сомножителей:

$$
\begin{array}{rl}
 & \times\ 0.\ 1\ 0\ 1\ 0 \quad -|[A_M]|_{ОК} \\
 & \quad 0.\ 1\ 0\ 0\ 1 \quad -|[B_M]|_{ОК} \\
\hline
 & \quad 0\ 1\ 0\ 1\ 0 \quad -\textit{1-е частичное произведение} \\
+ & \quad 0\ 0\ 0\ 0\ 0 \quad -\textit{2-е частичное произведение} \\
+ & \quad 0\ 0\ 0\ 0\ 0 \quad -\textit{3-е частичное произведение} \\
+ & \quad 0\ 1\ 0\ 1\ 0 \quad -\textit{4-е частичное произведение} \\
+ & \quad 0\ 0\ 0\ 0\ 0 \quad -\textit{5-е частичное произведение} \\
\hline
 & 0.\ 0\ 1\ 0\ 1\ 1\ 0\ 1\ 0 \quad -|[C_M^*]|_{ОК} = |[C_M^*]|_{ПК}
\end{array}
$$

Мантисса предварительного произведения ненормализованная, поэтому ее необходимо сдвинуть влево на один разряд, а предварительное значение порядка произведения уменьшить на единицу.

Таким образом, с учетом определенного знака, нормализации и округления до четырех разрядов конечное значение мантиссы $[C_M]_{ПК} = 1.1011$.

С учетом нормализации конечное значение порядка $[C_E]_{ПК} = 1.010$.

Конечное значение произведения C: $\{[C_E]_{ПК} = 1.010, [C_M]_{ПК} = 1.1011\}$.

Деление чисел с плавающей точкой (запятой)

С точки зрения представления чисел в форме с плавающей точкой поиск частного C двух таких чисел A и B сводится к поиску частного на основании порядка и мантиссы делимого и делителя. Для двоичных чисел это имеет вид

$$C = A : B = (2^{A_E} \cdot A_M) : (2^{B_E} \cdot B_M) = 2^{A_E - B_E} : (A_M \cdot B_M) = 2^{C_E} \cdot C_M. \quad (2.2)$$

Из формулы (2.2) следует, что порядок частного определяется как разность порядков делимого и делителя, а мантисса – как частное от деления мантиссы делимого на мантиссу делителя. Однако, учитывая то, что при делении мантисс может произойти нарушение нормализации, в результате указанных действий будут найдены предварительные значения порядка и мантиссы искомого частного. Окончательные значения порядка и мантиссы частного будут определены после устранения нарушения нормализации в предварительном результате.

Последовательность действий нахождения частного двух чисел, представленных в формате с плавающей точкой, имеет следующий общий вид:

– определяется знак частного (как знак мантиссы делимого *XOR* знак мантиссы делителя);

– предварительное значение порядка частного определяется как разность порядков делимого и делителя;

– предварительное значение мантиссы частного определяется как частное мантисс делимого и делителя (методом с/без восстановления остатка);

– устраняется нарушение нормализации мантиссы частного (если нарушение присутствует) соответствующей корректировкой предварительного значения порядка и мантиссы искомого частного;

– при формировании мантиссы частного нормализованных чисел с плавающей точкой возможен только один вид нарушения нормализации – нарушение нормализации слева от точки на один знак.

Деление мантисс выполняется по правилам деления чисел с фиксированной точкой с восстановлением остатка или без восстановления остатка.

Деление с восстановлением остатка выполняется потактно за (n + 2) такта (n – разрядность модульного поля представления делимого). На каждом такте определяется один разряд частного.

На каждом такте выполняются следующие действия:

1 Выполняется пробное вычитание:

1.1 На первом такте из делимого вычитается делитель (прибавляется с противоположным знаком);	1.2 На последующих тактах из полученных остатков вычитается делитель (прибавляется с противоположным знаком);

2 Анализируется знак остатка после вычитания:

2.1.1 Если знак отрицательный, то в очередном разряде формируемого частного устанавливается нуль; 2.1.2 Осуществляется восстановление остатка, т. е. к разности пробного вычитания прибавляется делитель;	2.2.1 Если знак положительный, то в очередном разряде формируемого частного устанавливается единица; 2.2.2 Восстановление остатка не осуществляется;

3 Выполняется умножение на два (арифметический сдвиг влево) полученного остатка (восстановленного или невосстановленного).

На первом такте определяется разряд целой части искомого частного. Для правильной дроби этот разряд должен иметь нулевое значение. Если на первом

такте будет получен единичный разряд целой части искомого частного, то вырабатывается специальный сигнал о том, что искомое частное не является правильной дробью.

Деление без восстановления остатка также выполняется за (n + 2) такта.

На первом такте выполняется пробное вычитание: из делимого по модулю вычитается делитель по модулю (прибавляется с отрицательным знаком). По знаку результата пробного вычитания определяется старший разряд частного (если результат положительный – единица, а если отрицательный – нуль). Старший разряд является разрядом целой части частного и, если он ненулевой, вырабатывается сигнал о нарушении нормализации (неправильная дробь). Далее выполняется арифметический сдвиг влево на один разряд – получается остаток.

На каждом последующем такте выполняются следующие действия:

1 Анализируется знак получаемых остатков:

1.1 Если знак положительный, то вычитается делитель (прибавляется с противоположным знаком);	1.2 Если знак отрицательный, то прибавляется делитель;

2 Анализируется знак полученной суммы:

2.1 Если знак положительный, то в очередном разряде формируемого частного устанавливается единица;	2.2 Если знак отрицательный, то в очередном разряде формируемого частного устанавливается нуль;

3 Выполняется арифметический сдвиг влево полученной суммы – получается новый остаток. Повторяются шаги 1 – 3.

После выполнения последнего (n + 2)-го такта и нормализации выполняется округление до нужного количества разрядов.

Пример

Найти частное C двух чисел A и B, представленных в формате с плавающей точкой, если эти числа представлены в виде порядков $[A_E]_{ПК}$ = 1.010 и $[B_E]_{ПК}$ = 0.001 соответственно и мантисс $[A_M]_{ПК}$ = 1.1010 и $[B_M]_{ПК}$ = 0.1001 соответственно. При выполнении операций использовать модифицированный дополнительный код. При делении мантисс использовать метод деления без восстановления остатка.

Сначала определяем знак искомого частного. Так как знаки мантисс делимого и делителя неодинаковые, знак частного будет отрицательным.

Предварительное значение порядка частного C_E^* определяется как разность порядков делимого и делителя $C_E^* = A_E - B_E$. Операция вычитания заменяется на операцию сложения с изменением знака вычитаемого на противоположный. Таким образом:

$$
\begin{array}{rll}
 & 1\,1.1\,1\,0 & -[A_E]_{МДК} \\
+ & 1\,1.1\,1\,1 & -[-B_E]_{МДК} \\
\hline
 & 1\,1.1\,0\,1 & \text{– переполнение знакового поля в МДК игнорируется} \\
 & 1\,1.1\,0\,1 & -[C_E^*]_{МДК} \\
 & 1\,1.0\,1\,1 & -[C_E^*]_{МПК}
\end{array}
$$

Абсолютное значение предварительного значения мантиссы частного C_M^* определяется за шесть тактов деления методом без восстановления остатка:

	$\dot{0}\ \dot{0}.\ \dot{1}\ \dot{0}\ 1\ 0$	– $[\lvert A_M\rvert]_{МДК}$
+	$1\ 1.\ 0\ 1\ 1\ 1$	– $[-\lvert B_M\rvert]_{МДК}$
	$\dot{1}\ 0\ 0.\ 0\ 0\ 0\ 1$	– *переполнение разрядности знакового поля в МДК игнорируется*
+	$0\ 0.\ 0\ 0\ 0\ 1$	– *положительный остаток* 1*-го такта:* **1*-й разряд частного* = 1**
	$0\ 0.\ \dot{0}\ \dot{0}\ 1\ 0$	– *остаток после арифметического сдвига влево*
+	$1\ 1.\ 0\ 1\ 1\ 1$	– $[-\lvert B_M\rvert]_{МДК}$
	$1\ 1.\ 1\ 0\ 0\ 1$	– *отрицательный остаток* 2*-го такта:* **2*-й разряд частного* = 0**
	$1\ 1.\ 0\ 0\ 1\ 0$	– *остаток после арифметического сдвига влево*
+	$0\ 0.\ 1\ 0\ 0\ 1$	– $[\lvert B_M\rvert]_{МДК}$
	$1\ 1.\ 1\ 0\ 1\ 1$	– *отрицательный остаток* 3*-го такта:* **3*-й разряд частного* = 0**
	$1\ 1.\ 0\ 1\ 1\ 0$	– *остаток после арифметического сдвига влево*
+	$0\ 0.\ 1\ 0\ 0\ 1$	– $[\lvert B_M\rvert]_{МДК}$
	$1\ 1.\ 1\ 1\ 1\ 1$	– *отрицательный остаток* 4*-го такта:* **4*-й разряд частного* = 0**
	$\dot{1}\ \dot{1}.\ 1\ 1\ 1\ 0$	– *остаток после арифметического сдвига влево*
+	$0\ 0.\ 1\ 0\ 0\ 1$	– $[\lvert B_M\rvert]_{МДК}$
	$\dot{1}\ 0\ 0.\ 0\ 1\ 1\ 1$	– *переполнение разрядности знакового поля в МДК игнорируется*
	$0\ 0.\ 0\ 1\ 1\ 1$	– *положительный остаток* 5*-го такта:* **5*-й разряд частного* = 1**
	$\dot{0}\ \dot{0}.\ \dot{1}\ \dot{1}\ 1\ 0$	– *остаток после арифметического сдвига влево*
+	$\dot{1}\ \dot{1}.\ 0\ 1\ 1\ 1$	– $[-\lvert B_M\rvert]_{МДК}$
	$\dot{1}\ 0\ 0.\ 0\ 1\ 0\ 1$	– *переполнение разрядности знакового поля в МДК игнорируется*
	$0\ 0.\ 0\ 1\ 0\ 1$	– *положительный остаток* 6*-го такта:* **6*-й разряд частного* = 1**
	$0\ 0.\ 1\ 0\ 1\ 0$	– *остаток после арифметического сдвига влево*

Таким образом, учитывая знаки остатков, полученных на шести тактах, абсолютное предварительное значение мантиссы искомого частного равно

$|C_M^*| = 1{,}00011$.

Мантисса частного не нормализованная (нарушение нормализации слева от точки, т. к. получена ненулевая целая часть), поэтому необходимо сдвинуть мантиссу вправо на один разряд, а предварительное значение порядка частного увеличить на единицу. После нормализации мантиссы (и корректировки порядка частного), ее округления до четырех разрядов в модуле, с учетом определенного ранее знака, окончательное значение C равно

C: $\{[C_E]_{ПК} = 1.010, [C_M]_{ПК} = 1.1001\}$.

2.7 Контрольные вопросы

1 Чему будет равна сумма двух единичных разрядов при сложении положительных двоичных чисел?

2 Как происходит формирование разности нулевого и единичного разрядов положительных двоичных чисел?

3 В чем заключается особенность формирования разности двух положительных двоичных чисел, если из разряда уменьшаемого происходил заем единицы в младший разряд?

4 В какую сторону происходит сдвиг частичных произведений при умножении двух положительных двоичных чисел, начиная с младшего разряда и начиная со старшего разряда?

5 Какое количество разрядов делимого используется на первом этапе деления двух положительных двоичных чисел?

6 При превышении какого значения тетрады необходима корректировка двоичной суммы при сложении в двоично-десятичной системе счисления?

7 На какое значение и почему происходит корректировка тетрад двоичной разности при вычитании в двоично-десятичной системе счисления?

8 Какие виды поразрядных операций существуют?

9 Какие бывают виды поразрядных сдвигов (влево или вправо)?

10 В чем заключается особенность выполнения поразрядного арифметического сдвига для беззнаковых и знаковых целых чисел?

11 Какой код отражает количественную оценку числа?

12 К каких кодах выполняются арифметические операции над алгебраическими двоичными числами?

13 Как реализуется операция вычитания для алгебраических двоичных чисел?

14 Как учитывается переполнение знакового поля при сложении в обратном и в дополнительном кодах?

15 Какой вид переполнения обрабатывается в первую очередь?

16 Опишите алгоритм сложения (вычитания) вещественных чисел в формате с фиксированной точкой.

17 Опишите алгоритм умножения вещественных чисел в формате с фиксированной точкой.

18 Опишите алгоритм деления вещественных чисел в формате с фиксированной точкой.

19 Какие способы деления мантисс вещественных чисел существуют?

20 Каким образом осуществляется выравнивание порядков при сложении вещественных чисел в формате с плавающей точкой (запятой)?

21 Каким образом выполняется устранение нарушения нормализации при выполнении арифметических операций над вещественными числами в формате с плавающей точкой (запятой)?

22 Каким образом осуществляется нормализация при умножении чисел, представленных в формате с плавающей точкой (запятой)?

23 Какой вид нарушения нормализации имеет место быть при умножении и какой при делении чисел, представленных в формате с плавающей точкой?

24 Как определяется знак результата при умножении или делении вещественных чисел в формате с плавающей точкой (запятой)?

25 Опишите алгоритм деления с восстановлением остатка вещественных чисел в формате с плавающей точкой (запятой).

26 Опишите алгоритм деления без восстановления остатка вещественных чисел в формате с плавающей точкой (запятой).

3 ПРИНЦИПЫ ПОСТРОЕНИЯ ЭЛЕКТРОННЫХ ВЫЧИСЛИТЕЛЬНЫХ МАШИН

3.1 Элементы электронных вычислительных машин

Элементы ЭВМ представляют собой простейшие радиотехнические детали, на основе которых строятся более крупные части ЭВМ – узлы. Общее условное графическое обозначение абстрактного элемента представлено на рисунке 3.1 и предполагает обозначение трех полей:

а) поля входов (слева). Количество входов n, где $n \geq 1$;

б) поля выходов (справа). Количество выходов m, где $m \geq 1$;

в) поле обозначения элемента (по центру), несущее информацию о типе элемента и, возможно, исполняемых им функций.

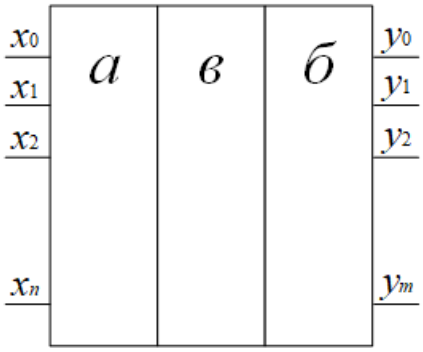


Рисунок 3.1 – Общее обозначение элемента ЭВМ

Элементы ЭВМ бывают трех основных разновидностей:

1 *Логические* – реализующие какие-либо логические функции алгебры логики.

2 *Запоминающие* – триггеры (автомат Мура), реализующие хранение одного из двух устойчивых состояний.

3 *Специальные (вспомогательные)* – усилители, формирователи и генераторы сигналов, преобразователи логических уровней, индикаторы и др.

3.2 Основные узлы электронных вычислительных машин

Все основные узлы ЭВМ строятся на базе элементов и, как правило, представляют собой совокупность нескольких логических элементов или элементов памяти.

Выделяются следующие основные узлы ЭВМ:

1 *Комбинационные* – выходные сигналы зависят только от действующих в настоящее время входных сигналов.

2 *Накапливающие* – выходные сигналы зависят не только от действующих в настоящее время входных сигналов, но и от поступавших ранее.

К комбинационным узлам относятся:

1 *Одноразрядный сумматор* – узел, выполняющий операцию поразрядного сложения над одним двоичным разрядом. Входами такого сумматора являются разряд первого слагаемого a, разряд второго слагаемого b, признак переноса из младшего разряда p, а выходами являются сумма в текущем разряде S и признак переноса в старший разряд P.

2 *Многоразрядный сумматор* – узел, построенный на базе одноразрядных сумматоров с подключением соответствующих выходов к соответствующим входам («*p*» к «*P*») и выполняющий сложение многоразрядных двоичных чисел. Многоразрядные сумматоры бывают с последовательным и со сквозным переносом.

3 *Сумматор по модулю* 2 – узел, реализующий поразрядное сложение по модулю 2 (логическую операцию исключающее ИЛИ).

4 *Шифратор (кодер)* – узел, преобразующий сигнал на одном из n входов в комбинацию сигналов на m выходах и использующийся в случае, когда необходимо передавать большое количество данных при небольшом количестве линий связи. При этом количество входов n и выходов m связано соотношением $2^m \geq n$. Также имеет вход синхронизации.

5 *Дешифратор (декодер)* – узел, преобразующий комбинацию сигналов на n входах в сигнал на одном из m выходов, т. е. с помощью n входов можно задавать выход, на который будет подаваться единичный сигнал. Количество входов n и выходов m связано соотношением $2^n \geq m$. Также имеет вход синхронизации. Дешифратор применяется для построения мультиплексора и демультиплексора.

6 *Мультиплексор* – узел, обеспечивающий подключение одного из $2^n + n$ своих входов ($x_0, x_1, \ldots, x_{2n-1}, s_0, s_1, \ldots, s_{n-1}$) к единственному выходу m, на который подается значение на входе x_i, где i – число, которое кодируется входами $s_0, s_1, \ldots, s_{n-1}$. Также имеет вход синхронизации. Используется при мультиплексированной передаче данных в шинных архитектурах связей (см. подраздел 3.3).

7 *Демультиплексор* – узел, обеспечивающий логическое подключение одного входа к одному из нескольких выходов, т. е. обратную мультиплексору функцию. Также имеет вход синхронизации.

Накапливающие узлы строятся на базе триггеров, каждый из которых предназначен для хранения отдельного разряда.

Триггеры бывают различных видов:

1 *RS*-триггер. Имеет два входа q_R (*reset*, сброс) и q_S (*set*, установка) и два выхода Q и $\bar{Q}$ (обратное значение выхода Q, т. е. инверсия разряда). Для установки триггера на значение «0» ($Q = 0$, $\bar{Q} = 1$) необходимо подать единичный сигнал на вход q_R, а для установки триггера в значение «1» ($Q = 1$, $\bar{Q} = 0$) необходимо подать единичный сигнал на вход q_S. Комбинация из двух единичных сигналов на обоих входах запрещена.

2 *T*-триггер. Имеет один вход q_T и два выхода Q и $\bar{Q}$. Единичный сигнал, поданный на вход q_T, переводит выходы *T*-триггер в противоположное состояние.

3 *JK*-триггер. Имеет два входа q_J и q_K и два выхода Q и $\bar{Q}$. Аналогичен *RS*-триггеру, только комбинация из двух единичных сигналов на обоих входах не запрещена и переводит выходы *JK*-триггер в противоположное состояние (аналогично *T*-триггеру).

4 *D*-триггер. Имеет один вход q_D (*digital*, цифра) и два выхода Q и $\bar{Q}$. Для установки триггера в значение «0» ($Q = 0$, $\bar{Q} = 1$) необходимо подать нулевой сигнал на вход q_D, а для установки триггера в значение «1» ($Q = 1$, $\bar{Q} = 0$) необходимо подать единичный сигнал на вход q_D. То есть триггер хранит значение, поданное на вход.

Зависимость хранимого триггерами значения от сигналов на входах наглядно показана в таблице 3.1.

Таблица 3.1 – Зависимость хранимого триггером значения от сигналов на входах

<table>
<tr><td colspan="4">RS-триггер</td><td colspan="3">T-триггер</td><td colspan="4">JK-триггер</td><td colspan="3">D-триггер</td></tr>
<tr><td colspan="2">Входы триггера</td><td colspan="2">Выходы триггера</td><td>Входы триггера</td><td colspan="2">Выходы триггера</td><td colspan="2">Входы триггера</td><td colspan="2">Выходы триггера</td><td>Входы триггера</td><td colspan="2">Выходы триггера</td></tr>
<tr><td>q_R</td><td>q_S</td><td>Q</td><td>$\bar{Q}$</td><td>q_T</td><td>Q</td><td>$\bar{Q}$</td><td>q_J</td><td>q_K</td><td>Q</td><td>$\bar{Q}$</td><td>q_D</td><td>Q</td><td>$\bar{Q}$</td></tr>
<tr><td>0</td><td>0</td><td colspan="2">Хранение текущих значений Q и $\bar{Q}$</td><td rowspan="2">0</td><td colspan="2" rowspan="2">Хранение текущих значений Q и $\bar{Q}$</td><td>0</td><td>0</td><td>0</td><td>1</td><td rowspan="2">0</td><td rowspan="2">0</td><td rowspan="2">1</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td><td>0</td><td>1</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>0</td><td rowspan="2">1</td><td colspan="2" rowspan="2">Инверсия текущих значений Q и $\bar{Q}$</td><td>0</td><td>1</td><td>1</td><td>0</td><td rowspan="2">1</td><td rowspan="2">1</td><td rowspan="2">0</td></tr>
<tr><td>1</td><td>1</td><td colspan="2">Недопустимая комбинация q_R и q_S</td><td>1</td><td>1</td><td colspan="2">Инверсия текущих значений Q и $\bar{Q}$</td></tr>
</table>

К <u>накапливающим узлам</u> относятся:

1 *Регистр* – узел, предназначенный для хранения многоразрядных значений (отдельного числа).

Регистр – группа триггеров, количество которых соответствует количеству разрядов в хранимом регистром двоичном числе.

Разрядность регистра определяется архитектурой ЭВМ.

Машинное слово – двоичное число, хранимое в регистре.

Для упрощения доступа к регистрам они имеют уникальные номера. Этот уникальный номер называется *адресом регистра*.

Совокупность всех регистров основной памяти – это *оперативная память*.

2 *Счетчик* – узел, предназначенный для хранения, как правило, многоразрядных значений, который по каждому сигналу изменяет (увеличивает на единицу) хранимый код числа.

Счетчики могут работать в двух режимах: в режиме счетчика событий и в режиме таймера.

В режиме счетчика происходит увеличение на единицу хранимого значения по каждому единичному сигналу, поступающему на вход счетчика по факту наступления какого-либо события.

В режиме таймера счетчик считает тактовые импульсы процессора или импульсную последовательность с выхода управляемого делителя частоты (например, для счета времени).

Существует множество различных счетчиков/таймеров, каждый из которых в ЭВМ выполняет свою функцию, например:

– счетчик команд – служит для хранения адреса ячейки оперативной памяти (по сути регистра) текущей команды, выполняемой процессором;

– сторожевой таймер – предназначен для защиты микропроцессорной системы (МПС) от зависания программы. При запуске на выполнение программы сторожевой таймер начинает считать тактовые импульсы. Сторожевой таймер сбрасывается на программном уровне, т. е. при зависании программы на аппаратном уровне еще продолжится счет тактов и при достижении счетчиком максимального кода (в зависимости от разрядности) генерируется сигнал внутреннего сброса (внутреннее прерывание), перезагружающий МПС.

3.3 Принципы построения электронных вычислительных машин

В 1945 году Джоном фон Нейманом были сформулированы основополагающие принципы построения ЭВМ, которые позже реализовались на практике, и такие машины получили название фоннеймановских (или на фоннеймановской архитектуре).

Принципы фон Неймана:

1 Принцип двоичного кодирования

Вся информация – как данные, так и команды – кодируется двоичными цифрами «0» и «1». Каждый тип информации представляется двоичной последовательностью и имеет свой формат, который составляется из участков разной длины. Каждый участок последовательности битов в формате имеет определенный смысл и называется полем. В числовой информации обычно выделяют поле знака и поле модуля числа. В формате команды обычно выделяют два основных поля: поле кода операции и поле адресов. Код операции представляет собой указание, какая операция должна быть выполнена, и задается с помощью *r*-разрядной двоичной комбинации.

Вид адресной части и число составляющих ее адресов зависят от типа команды: в командах преобразования данных адресное поле содержит адреса операндов и результата; в командах изменения порядка вычислений – адрес следующей команды программы; в командах ввода/вывода – номер устройства.

2 Принцип программного управления

Все вычисления, предусмотренные алгоритмом решения задачи, должны быть представлены в виде программы, состоящей из последовательности управ-

ляющих слов-команд. Команды программы хранятся в последовательных ячейках памяти вычислительной машины и выполняются в естественной последовательности, т. е. в порядке их следования в программе.

3 Принцип однородности памяти

Команды и данные могут храниться в одной и той же памяти. Циклические изменения адресной части команды обеспечивают обращение к последовательным элементам массива данных. Это носит название *модификации команд*.

Другой подход принципа однородности, когда команды одной программы могут быть получены как результат исполнения другой программы. Эта возможность лежит в основе трансляции программы – перевода исходного кода с языка высокого уровня на машинный язык конкретной ЭВМ.

4 Принцип адресности

Основная память (ОП) состоит из пронумерованных ячеек, доступных процессору в произвольный момент времени. Двоичные коды команд называются *словами* и хранятся в ячейках памяти, для доступа к которым используются номера соответствующих ячеек – *адреса*. Команды и данные должны располагаться в ОП так, что каждое слово хранится в отдельной ячейке, определяемой адресом, и соседние ячейки памяти имеют следующие по порядку адреса. Доступ к любым ячейкам ОП может производиться в произвольной последовательности (память с произвольным доступом), т. е. реализуется возможность условного перехода в процессе выполнения программы.

Адресность памяти позволяет использование условного перехода.

Возможность условного перехода в процессе выполнения программы позволяет строить программы различной конструкции (не только линейные). Таким образом, порядок выполнения команд однозначно задается программой.

Следствие введения фоннеймановских принципов:

- программа перестала быть постоянной частью машины;
- программу стало возможно легко изменить.

3.4 Классификация архитектур электронных вычислительных машин

Практически любая микропроцессорная система (МПС) имеет в своем составе следующие устройства:

1 *Процессор* – устройство, предназначенное для выполнения машинных команд, управления вычислительным процессом, его выполнением по программе и для координации работы всех других устройств МПС.

2 *Память* (в общем виде) – физическое устройство хранения данных.

3 *Устройства ввода/вывода* – устройства, предназначенные для взаимодействия МПС с внешней средой, в том числе с пользователем (человеком) и реализующие ввод данных в МПС и/или вывод данных пользователю, другой МПС или устройству.

Архитектура – это принцип построения чего-либо.

С точки зрения МПС, в частности ЭВМ, существует множество классификаций архитектур, таких как:

– архитектура связи устройств ЭВМ;

– архитектура ЭВМ по способам расположения команд и данных;

– архитектура ЭВМ по месту хранения операндов и др.

Архитектура связи устройств ЭВМ

Классическая структура связей (архаическая, с момента построения ЭВМ и до 1960-х гг.) и шинная (современная) представлена на рисунке 3.2.

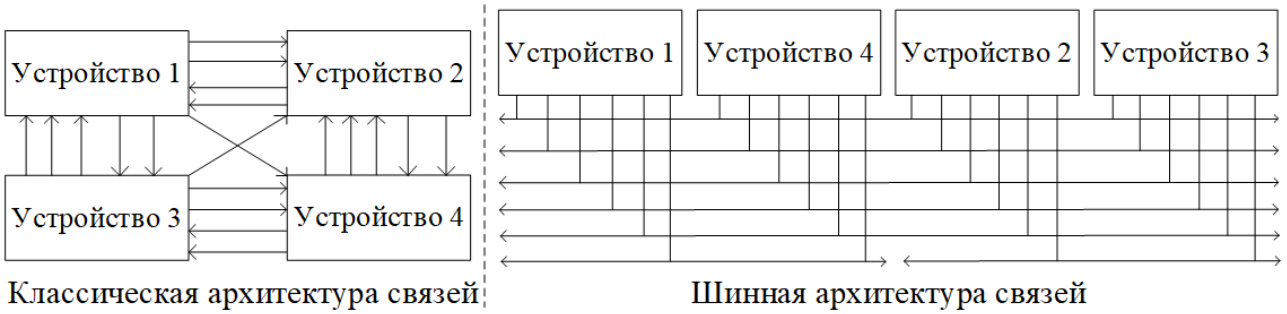


Рисунок 3.2 – Архитектуры связей устройств ЭВМ

При классической архитектуре связей все устройства, входящие в состав ЭВМ, соединены друг с другом независимо, по отдельным физическим линиям связи.

Достоинства классической архитектуры:

– независимость друг от друга устройств, входящих в МПС, при обмене информацией;

– относительно более высокая надежность работ МПС.

Недостатки классической архитектуры:

– множество линий связи и протоколов обмена информацией;

– громоздкость МПС из-за большого количества линий связи.

При шинной архитектуре связей все сигналы между всеми устройствами передаются по одним и тем же линиям связи, но в разное время (*мультиплексированная передача данных*). Причем передача по всем линиям связи может осуществляться в обоих направлениях (*двунаправленная передача данных*).

Достоинства шинной архитектуры:

– все устройства, подключенные к шине, имеют один протокол передачи информации (как следствие – унификация устройств, входящих в МПС);

– существенная компактность системы.

Недостатки шинной архитектуры:

– поскольку все устройства подключаются к каждой линии связи параллельно, неисправность любого устройства может вывести из строя всю МПС, если неисправное устройство портит линию связи (например, выдает информационные сигналы в шину в неположенное время);

– более низкое быстродействие, обусловленное последовательной по времени передачей информации между устройствами МПС.

Несмотря на все имеющиеся недостатки шинной архитектуры, решающим фактором ее применения послужила компактность такой МПС.

Структура системной шины (называемой *магистралью*) представлена на рисунке 3.3.

В состав системной шины (магистрали) входят шины более низкого уровня: шина данных, шина адреса, шина управления и шина питания. Каждое устройство, входящее в состав МПС, подключено к каждой из этих шин параллельно.

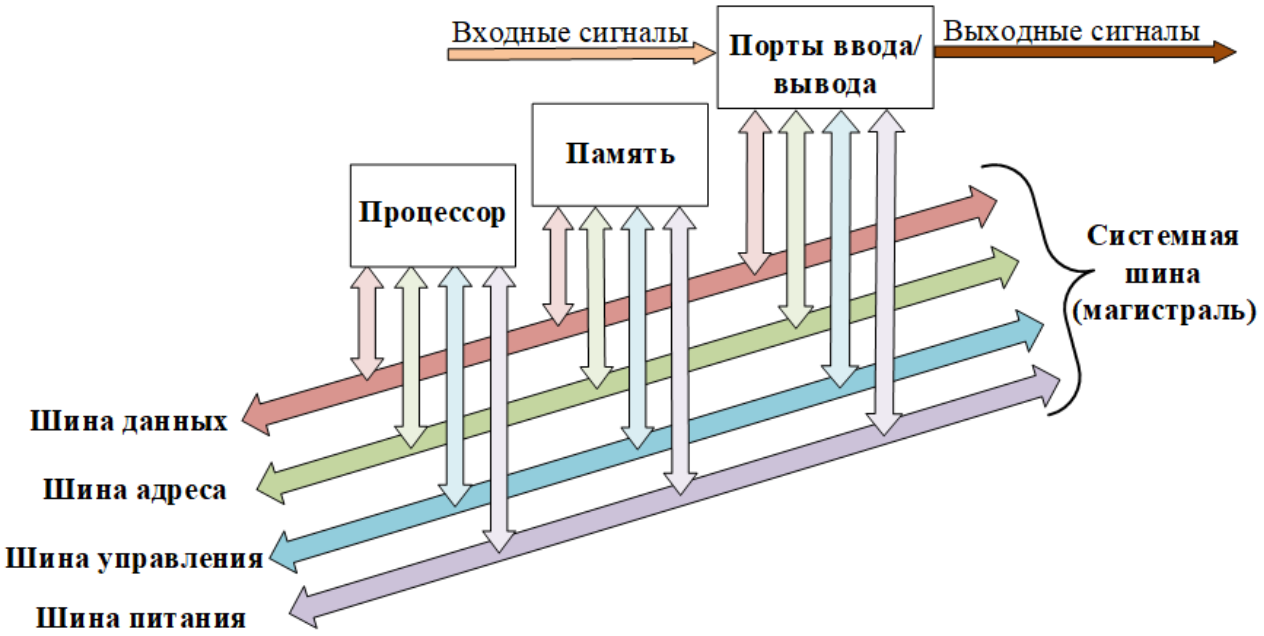


Рисунок 3.3 – Состав системной шины

Шина данных (*Data Bus*) – это основная шина, которая используется для передачи информационных кодов между всеми устройствами МПС. Возможна передача информации между устройствами даже без участия процессора, т. е. в режиме прямого доступа к памяти (*DMA*). Шина данных всегда двунаправленная.

Шина адреса (*Address Bus*) служит для определения адреса (номера) устройства, с которым процессор обменивается информацией в данный момент. Каждому устройству (кроме процессора) и каждой ячейке памяти в МПС присваивается собственный адрес. Шина адреса может быть однонаправленной или двунаправленной.

Шина управления (*Control Bus)* в отличие от шины адреса и шины данных передает отдельные управляющие сигналы. Каждый из них имеет свою функцию: некоторые сигналы служат для стробирования передаваемых или принимаемых данных (т. е. определяют моменты времени, когда информационный код выставлен на шину данных), другие управляющие сигналы используются для подтверждения приема данных, третьи – для сброса всех устройств в исходное состояние, четвертые – для тактирования всех устройств и т. д. Шина управления может быть однонаправленной или двунаправленной.

Шина питания (*Power Bus*) предназначена не для пересылки информационных сигналов, а для питания системы. Она состоит из линий питания и общего провода. В МПС может быть один или несколько источников питания (разного напряжения). Каждому напряжению питания соответствует своя линия связи. Все устройства подключены к этим линиям параллельно.

Архитектура ЭВМ по способам расположения команд и данных

По данной классификации существует Принстонская (одношинная, фоннеймановская) и Гарвардская (двухшинная) архитектуры (рисунок 3.4), причем обе подразумевают использование шинной архитектуры связей.

Принстонская архитектура реализует принцип совместного хранения команд (программы) и данных в памяти компьютера (т. е. в едином пространстве памяти), соединенной с остальными устройствами только одной шиной данных.

Гарвардская архитектура предполагает разделение памяти на память команд (программы) (ПК) и память данных (ПД), каждая из которых соединена с остальными устройствами отдельной шиной данных. То есть в состав системной шины при Гарвардской архитектуре входят две шины данных, названных по типу памяти: шина команд и шина данных.

Следует обратить внимание, что несмотря на неочевидность названия, обе шины (шина данных и шина команд) являются шинами данных (*data bus*).

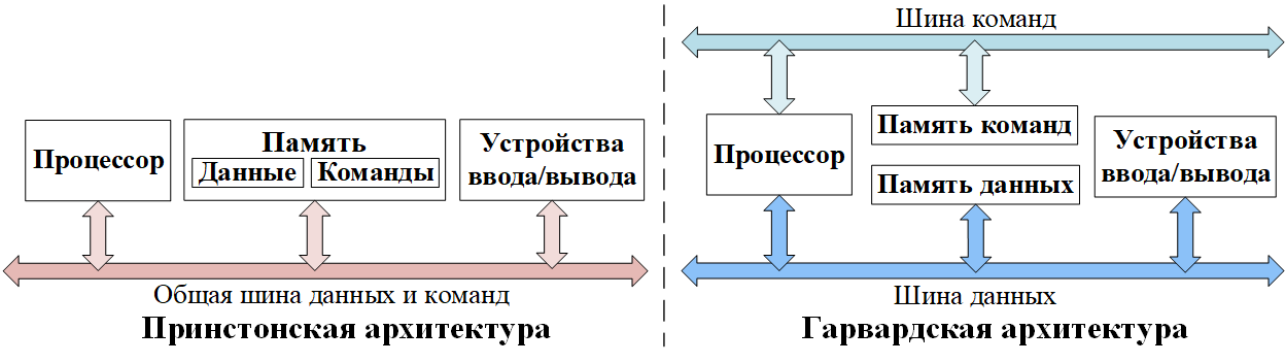


Рисунок 3.4 – Архитектуры ЭВМ по способам расположения команд и данных

Следует отметить, что данные архитектуры были разработаны в 1960-х гг. и в чистом виде давно устарели. В настоящее время на их основе разработаны более современные архитектуры, применяемые в ЭВМ, а именно:

– модифицированная Гарвардская архитектура (МГА), призванная устранить существенный недостаток Гарвардской архитектуры – большое количество интерфейсных выводов (ввиду наличия двух шин данных и адреса), что приводит к увеличению размеров и соответственно стоимости кристалла, на котором выполнен микропроцессор. Поэтому МГА имеет общую шину данных и шину адреса для всех внешних данных, а внутри процессора используется шина данных, шина команд и две шины адреса. Разделение шин в МГА осуществляется при помощи раздельных управляющих сигналов: чтения, записи или выбора области памяти. МГА нашла применение в сигнальных процессорах и микроконтроллерах;

– расширенная Гарвардская архитектура (РГА), предполагает использование кеш-памяти вместе с разделенными шинами данных (ПД + ПК + кеш), что позволяет за один такт получать из разной памяти команду на исполнение и два операнда. РГА нашла применение в цифровой обработке сигналов;

– различные гибридные модификации архитектур, например, процессорное ядро аппаратно является Гарвардским, но программно используется как Принстонское, что позволило одновременно сочетать в себе преимущества обеих систем: за один такт можно получать одновременно и команду, и операнды для ее выполнения (Гарвардская), но при этом написание программы происходит проще, как для Принстонской архитектуры. Применение: CISC-процессоры.

Архитектура ЭВМ по месту хранения операндов

Место хранения операндов и способ доступа к ним играет важную роль при выполнении команд процессором.

С этой точки зрения различают:

– стековую архитектуру;

– аккумуляторную архитектуру;

– регистровую архитектуру.

Использование той или иной архитектуры влияет на количество адресов в адресной части команд, их длину, простоту получения операндов и в конечном счете на общую длину команды, а соответственно, и скорость ее выполнения.

Стековая архитектура

Стек – это отличная от оперативной памяти структура организации данных, работающая по принципу «последним вошел – первым вышел» (*Last In – First Out, LIFO*). Стек можно представить, как показано на рисунке 3.5, в виде трубки, запаянной с одного конца, при этом через второй конец, называемый вершиной стека, происходит добавление и извлечение данных.

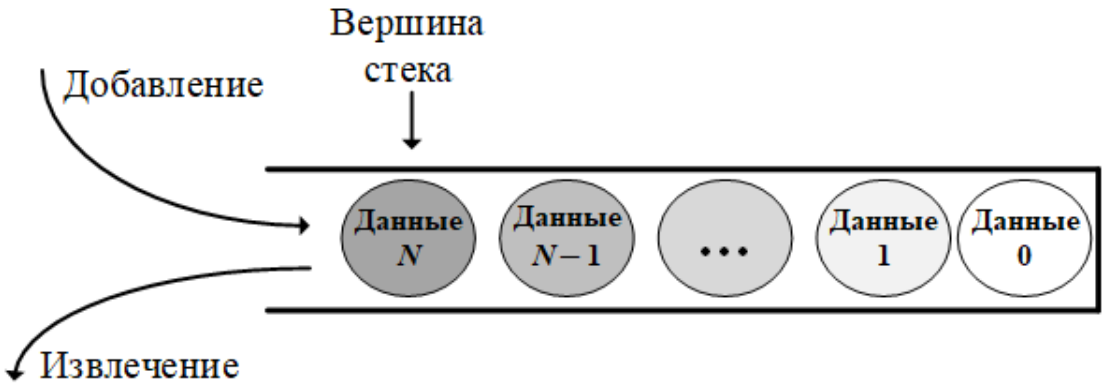


Рисунок 3.5 – Представление стека

Информация может быть занесена в вершину стека из памяти или из арифметико-логического устройства (АЛУ). Для записи в стек содержимого ячейки памяти с адресом *x* выполняется команда *push x*, по которой информация считывается из ячейки памяти, заносится в регистр данных, а затем проталкивается в стек. Результат операции из АЛУ заносится в вершину стека автоматически.

Сохранение содержимого вершины стека в ячейке памяти с адресом *x* производится командой *pop x*. По этой команде содержимое верхней ячейки стека подается на шину, с которой и производится запись в ячейку *x*, после чего вся находящаяся в стеке информация проталкивается на одну позицию вверх.

Для выполнения арифметической или логической операции на вход АЛУ подается информация, считанная из двух верхних ячеек стека (при этом содержимое стека продвигается на две позиции вверх, т. е. операнды из стека удаляются). Результат операции заталкивается в вершину стека или сразу же автоматически переписывается в память с помощью операции *pop x* (рисунок 3.6).

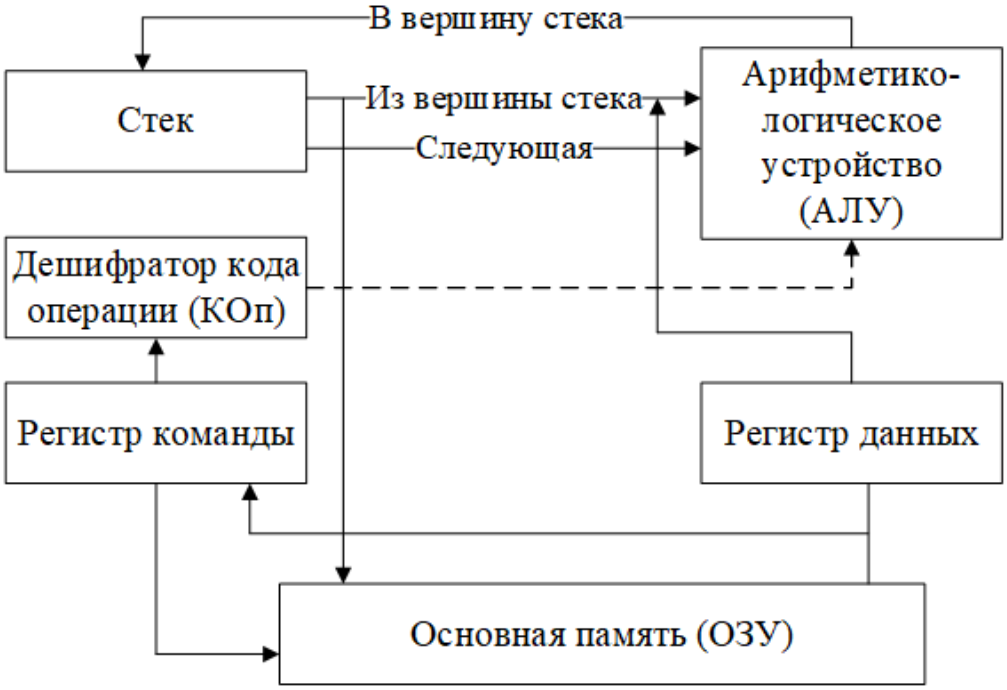


Рисунок 3.6 – Стековая архитектура

Аккумуляторная архитектура

При такой архитектуре хранение одного из операндов арифметической или логической операции предполагается в специальном регистре процессора – аккумуляторе, что предопределяет адрес нахождения этого операнда и избавляет от необходимости указывать его в команде. После выполнения операции в аккумулятор будет записан результат ее выполнения (рисунок 3.7).

Однако изначально оба операнда хранятся в основной памяти, и до выполнения операции один из них нужно загрузить в аккумулятор. После выполнения команды обработки результат находится в аккумуляторе и, если он не является операндом для последующей команды, его также требуется сохранить в основной памяти.

Для загрузки в аккумулятор содержимого ячейки памяти *x* предусмотрена команда загрузки *load x*. Запись содержимого аккумулятора в ячейку памяти *x* выполняется командой сохранения *store x*.

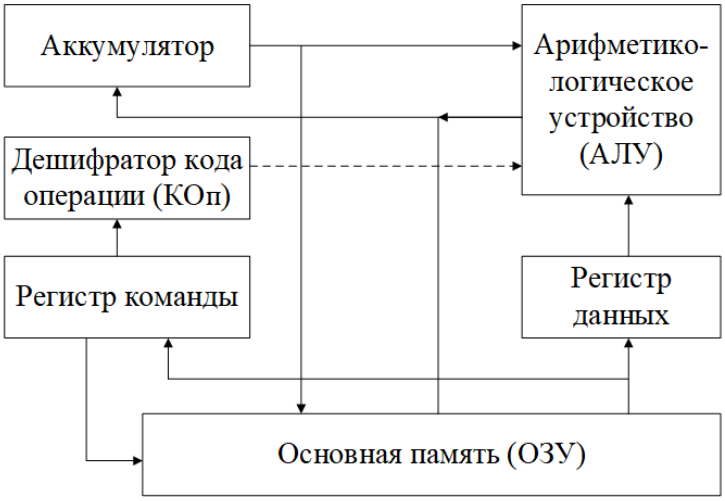


Рисунок 3.7 – Аккумуляторная архитектура

Регистровая архитектура

Регистровая архитектура допускает расположение операндов или в основной памяти, или в массиве регистров общего назначения (РОН). РОН можно рассматривать как явно управляемый кеш для хранения недавно использовавшихся данных (рисунок 3.8).

При таком размещении операндов возможны три варианта обработки:

– регистр – регистр: операнды могут находиться только в регистрах, и в них же записывается и результат;

– регистр – память: один из операндов размещается в регистре, а второй – в основной памяти. Результат обычно замещает один из операндов;

– память – память: оба операнда хранятся в основной памяти. Результат заносится в память.

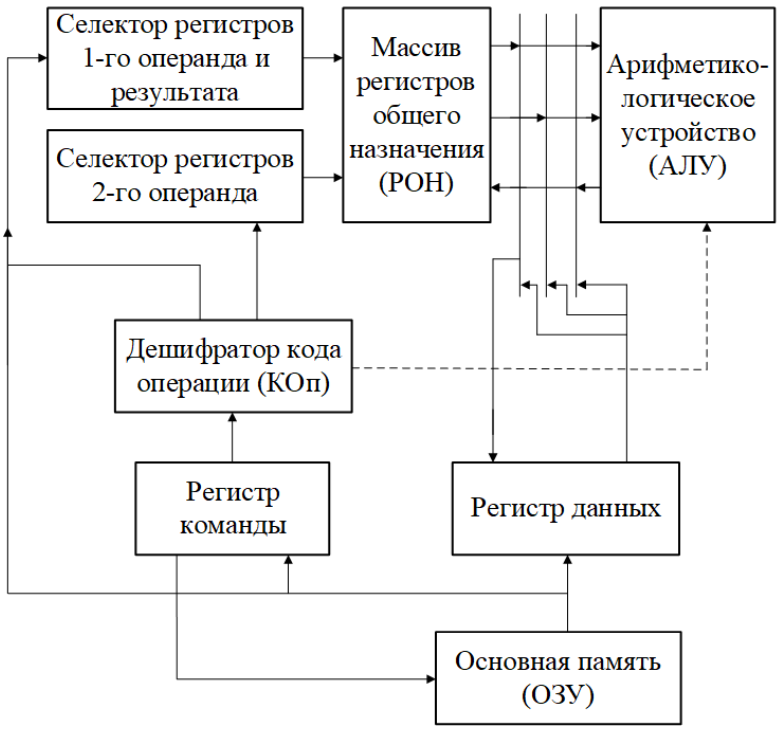


Рисунок 3.8 – Регистровая архитектура

Каждый из вариантов имеет свои достоинства и недостатки (таблица 3.2).

Таблица 3.2 – Достоинства и недостатки вариантов регистровой архитектуры

Вариант	Достоинства	Недостатки
Регистр – регистр	– Простота реализации; – фиксированная длина команд; – простая модель формирования объектного кода при компиляции программ; – возможность выполнения всех команд за одинаковое количество тактов	– Большая длина объектного кода; – из-за фиксированной длины команд часть разрядов в коротких командах не используется
Регистр – память	– Данные могут быть доступны без загрузки в регистры процессора; – простота кодирования команд; – объектный код получается достаточно компактным	– Потеря одного из операндов при записи результата; – длинное поле адреса памяти в коде команды сокращает место под номер регистра, что ограничивает общее число РОН. – количество тактов процессора, приходящихся на одну команду (*CPI*), зависит от места размещения
Память – память	– Компактность объектного кода; – малая потребность в регистрах для хранения промежуточных данных	– Разнообразие форматов команд и времени их исполнения; – низкое быстродействие из-за обращения к памяти

3.5 Система команд процессора и разновидности архитектур

Система команд – это набор допустимых для данного процессора управляющих кодов и способов адресации данных. В общем случае система команд жестко связана с конкретным типом процессора, поскольку определяется аппаратной структурой блока дешифрации команд. Поэтому разные процессоры могут иметь как разные команды в своей системе команд, так и их разное количество.

Тем не менее все команды системы команд процессора разбиты на различные общепринятые группы:

– пересылки данных (наиболее многочисленная группа). В таких командах содержится информация о адресе источника и получателя операндов (адреса ячеек памяти, номера регистров процессора или информация о том, что операнды расположены в стеке), длина подлежащих пересылке данных, заданная явно или косвенно, и способ адресации каждого из операндов, с помощью которого содержимое адресной части команды может быть пересчитано в физический адрес операнда;

– арифметической и логической обработки – команды, обеспечивающие арифметическую и логическую обработку информации в различных формах ее представления;

– ввода/вывода – это команды управления периферийным устройством, проверки состояния ввода/вывода и непосредственно команды ввода и вывода;

– управления потоком команд – это команды, способные изменить последовательность вычислений (безусловный переход), условный переход (ветвление), вызовы процедур и возвраты из процедур;

– битового управления – команды для реализации возможности независимого управления разрядами портов или регистров;

– преобразования и др.

Классификация систем команд

С точки зрения системы команд существуют следующие архитектуры:

– *CISC* (*Complicated Instruction Set Computer*) – полная система команд;

– *RISC* (*Reduced Instruction Set Computer*) – сокращенная система команд.

***CISC*-архитектура**

Характеристики:

– максимально полный набор команд, которые процессор может реализовывать. Такая система команд составляет сотни команд;

– команды имеют нефиксированный размер, кратный байту (1–8 байт);

– система команд, как правило, неортогональна, т. е. не все команды могут использовать любой из способов адресации применительно к любому из регистров процессора;

– выборка команды на исполнение осуществляется побайтно в течение нескольких циклов. Время выполнения команды может составлять от 1 до 12 циклов;

– обладает сложной кодировкой команд.

Достоинства:

– компактность наборов инструкций уменьшает размер программ и уменьшает количество обращений к памяти;

– наборы инструкций включают поддержку конструкций высокоуровневого программирования.

Недостатки:

– нерегулярность потока команд;

– высокая стоимость аппаратной части;

– сложности с распараллеливанием вычислений.

***RISC*-архитектура**

Характеристики:

– набор исполняемых команд сокращен до минимума. Такая система команд составляет десятки команд. Для реализации более сложных операций приходится комбинировать команды;

– все команды имеют формат фиксированной длины (не обязательно кратный байту);

– простая кодировка команд;

– выборка команды из памяти и ее исполнение осуществляется за один такт синхронизации;

– система команд предполагает возможность равноправного использования всех регистров процессора. Это обеспечивает дополнительную гибкость при выполнении ряда операций.

Достоинства:

– высокая скорость выполнения команд;

– меньшая необходимая площадь кристалла самого процессора, а следовательно, и более низкая стоимость;

– снижение нерегулярности потока команд.

Недостаток: для выполнения более сложных команд (отсутствующих в системе команд) требуется комбинирование нескольких простых команд.

Примечание – Время выполнения всей программы зависит не только от количества команд в программе, но и от того, какие именно команды используются. Так, однозначно нельзя сказать, какая из архитектур (*CISC* или *RISC*) будет производительнее. Сравнить можно только на примере конкретной выполняемой программы.

3.6 Форматы команд и способы адресации

Формат команды определяет ее структуру, т. е. количество разрядов, отводимых под всю команду, а также количество, разрядность и расположение отдельных полей команды

Поле команды – это совокупность разрядов, кодирующих составную часть команды.

Длина команды влияет на организацию и емкость памяти, структуру шин и быстродействие процессора.

Наиболее классическое представление формата команды содержит три части (рисунок 3.9).

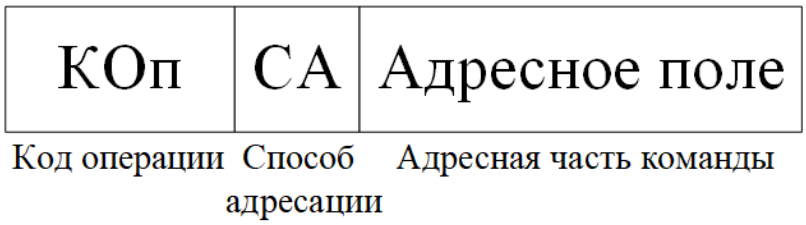


Рисунок 3.9 – Представление формата команды

Длина команды может быть определена как

$$R_{\text{К}} = \sum_{i=1}^{l} R_{\text{A}_i} + R_{\text{КОп}} + R_{\text{СА}}, \tag{3.1}$$

где l – количество адресов в адресном поле команды (адресность);

$R_{\text{A}i}$ – количество разрядов для записи i-го адреса;

$R_{\text{КОп}}$ – разрядность поля кода операции;

$R_{\text{СА}}$ – разрядность поля способа адресации.

Количество двоичных разрядов, отводимых под код операции, выбирается таким образом, чтобы каждая команда в системе команд имела уникальный номер. Если система команд предполагает N различных команд, то минимальная разрядность поля кода операции $R_{\text{КОпmin}}$ определяется как

$$R_{\text{КОпmin}} = \text{int}(\log_2 N_{\text{КОп}}). \qquad (3.2)$$

Адресность – это количество адресов в адресном поле команды.

В фоннеймановских машинах максимальная адресность команды может быть равна трем, а минимальная – единице (или при неявной адресации – нулю).

Трехадресный формат представлен на рисунке 3.10, *а*.

Если результат будет записан на место одного из операндов, то можно получить двухадресный формат команды (рисунок 3.10, *б*). В этом случае соответствующий замещаемый операнд после выполнения операции теряется.

Если один из операндов берется из регистра или стека или выполняется одноместная операция (операция над одним операндом), то можно сократить формат команды до одноадресного (рисунок 3.10, *в*).

Возможен также вариант, когда в адресном поле отсутствует адресная информация, при этом говорят о неявной адресации (например, либо адресного поля просто нет, либо отсутствующий адрес подразумевается кодом операции).

Операция	Адресная часть		
Код операции	1-й операнд	2-й операнд	Результат

а

Операция	Адресная часть	
Код операции	1-й операнд	2-й операнд, результат

б

Операция	Адресная часть
Код операции	1-й или 2-й операнд

в

а – трехадресный формат; *б* – двухадресный формат;
в – одноадресный формат

Рисунок 3.10 – Адресность формата команды

Исполнительный адрес операнда ($A_{исп}$) – это двоичный код номера ячейки памяти, служащий источником или приемником операнда. По этому коду происходит фактическое обращение к указанной ячейке.

Адресный код команды ($A_к$) – это двоичный код в адресном поле команды, из которого формируется исполнительный адрес операнда.

Способ адресации – это способ формирования исполнительного адреса операнда $A_{исп}$ по адресному коду команды $A_к$.

Способы адресации

Непосредственная адресация (НА)

При таком способе адресации в адресном поле команды вместо адреса содержится непосредственно само значение операнда (рисунок 3.11, *а*).

Этот способ может применяться при выполнении арифметических операций, операций сравнения, а также для загрузки констант в регистры. Когда операндом является число, оно обычно представляется в дополнительном коде. При записи в регистр, имеющий разрядность, превышающую длину непосредственного операнда, операнд размещается в младшей части регистра, а оставшиеся свободными позиции заполняются значением знакового бита операнда.

Прямая адресация (ПА)

При прямой адресации адресный код прямо указывает номер ячейки памяти, к которой производится обращение, т. е. адресный код команды $A_к$ = исполнительному адресу операнда $A_{исп}$ (рисунок 3.11, *б*).

Недостаток данного способа адресации – ограниченный размер адресного поля команды, т. к. для адресации к памяти большой емкости нужно большое количество разрядов адресного поля.

Косвенная адресация (КА)

При косвенной адресации с помощью ограниченного адресного поля команды указывается адрес ячейки небольшого адресуемого пространства памяти, содержащей, в свою очередь, полноразрядный адрес операнда (рисунок 3.11, *в*).

При косвенной адресации содержимое адресного поля команды остается неизменным, в то время как косвенный адрес в процессе выполнения программы можно изменять. Это позволяет проводить вычисления, когда адреса операндов заранее неизвестны и появляются лишь в процессе решения задачи. Такой прием упрощает передачу параметров подпрограммам и обработку массивов и списков.

Регистровая адресация (РА)

При регистровой адресации адресное поле команды содержит не адрес ячейки памяти, а адрес регистра процессора, по которому находится операнд (рисунок 3.11, *г*). Обычно размер адресного поля в данном случае составляет 4–6 бит, что позволяет указать на один из 16–64 регистров общего назначения (РОН). Основными преимуществами регистровой адресации являются короткое адресное поле в команде и исключение обращений к памяти. Возможности использования регистровой адресации ограничены числом РОН в составе процессора.

Относительная адресация (ОА)

При относительной адресации для получения исполнительного адреса операнда необходимо адресный код команды сложить со значением содержимого счетчика команд (рисунок 3.11, *д*).

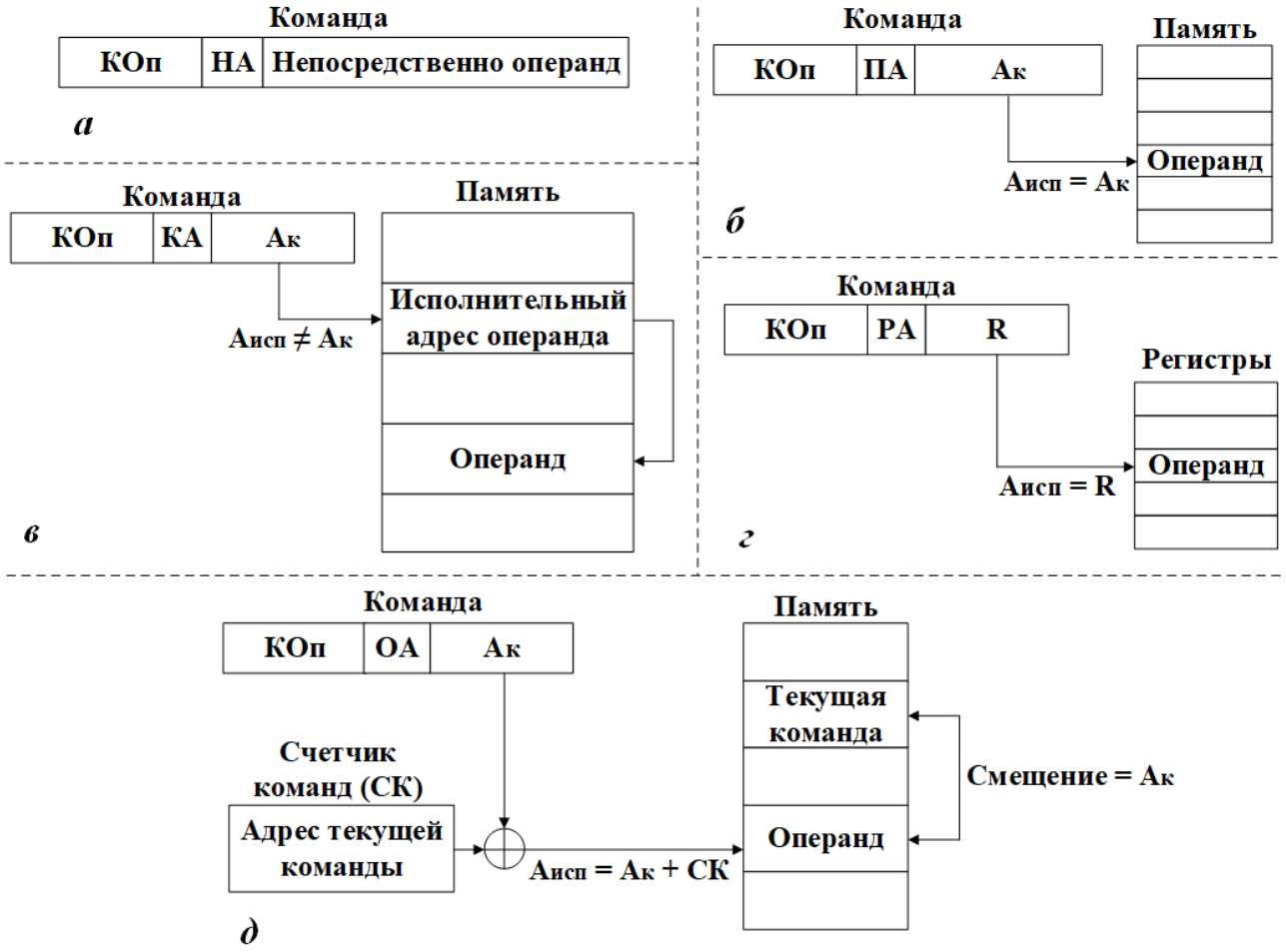


Рисунок 3.11 – Способы адресации

Таким образом, адресный код команды представляет собой смещение относительно адреса текущей команды. Однако в момент вычисления исполнительного адреса операнда в счетчике команд может уже быть сформирован адрес следующей команды, что нужно учитывать при выборе величины смещения.

Поскольку смещение будет складываться со значением счетчика команд, а операция сложения не выполняется в прямом коде, то чаще всего адресный код команды уже записан в дополнительном коде.

3.7 Основные устройства процессора

Современные процессоры в своем составе имеют такие устройства, как устройство управления (УУ), арифметико-логическое устройство (АЛУ), наборы различных регистров (РОН, регистры специальных функций, аккумуляторы, блок управляющих регистров и т. д.), контроллеры и кеш-память.

Простейшая структура процессора представлена на рисунке 3.12.

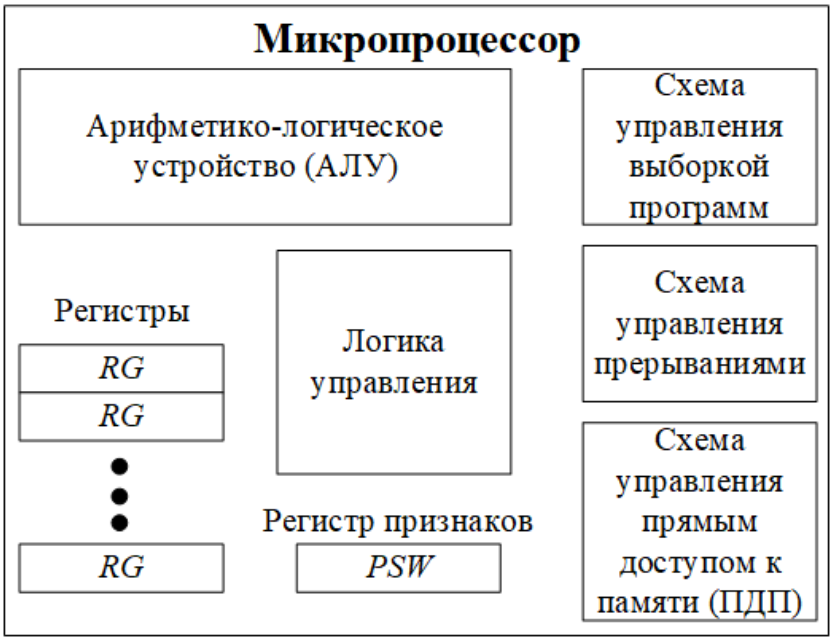


Рисунок 3.12 – Простейшая структура процессора

УУ организует автоматическое выполнение программ. Пересылка информации между любыми элементами ЭВМ инициируется своим сигналом управления (СУ), т. е. управление вычислительным процессом сводится к выдаче нужного набора СУ в нужной временной последовательности. Основной функцией УУ является формирование управляющих сигналов, отвечающих за извлечение команд из памяти в порядке, определяемом программой, и последующее исполнение этих команд. Кроме того, УУ формирует СУ для синхронизации взаимодействия внутренних и внешних устройств ЭВМ.

АЛУ обеспечивает арифметическую и логическую обработку двух входных операндов, в результате чего формируется значение выходной переменной. Функции АЛУ обычно сводятся к простым арифметическим и логическим операциям, а также операциям сдвига.

Помимо вычисления результата арифметические и логические операции сопровождаются формированием в АЛУ признаков (флагов), характеризующих этот результат. Наиболее часто фиксируются такие признаки, как *Z* (*Zero*) – нулевой результат, *N* (*Negative*) – отрицательный результат, *V* (*Overflow*) – переполнение разрядной сетки, *C* (*Carry*) – наличие переноса.

В процессоре может быть одно универсальное АЛУ для выполнения всех основных арифметических и логических преобразований или несколько специализированных АЛУ или операционных блоков для отдельных видов операций, например, АЛУ для чисел с фиксированной точкой и АЛУ для чисел с плавающей точкой.

Блок управляющих регистров предназначен для временного хранения управляющей информации. Он содержит регистры и счетчики, участвующие в управлении вычислительным процессом: регистры, хранящие информацию о состоянии процессора, регистр-счетчик адреса команды – счетчик команд, счетчики тактов, регистр запросов прерывания и др.

3.8 Контрольные вопросы

1 Какие бывают виды основных узлов ЭВМ?

2 Какие узлы относятся к комбинационным?

3 Какие виды узлов относятся к накапливающим?

4 В чем заключается отличие *RS*- от *JK*-триггера?

5 Что такое регистр?

6 В каких режимах может работать счетчик?

7 Назовите принципы построения вычислительной машины фон Неймана. Какой из принципов можно назвать наиболее существенным?

8 Что включает в себя архитектура фоннеймановской вычислительной машины?

9 Какие виды архитектур по месту хранения операндов существуют?

10 Какие варианты регистровой архитектуры существуют?

11 Назовите базовые принципы *CISC*-архитектуры.

12 Назовите базовые принципы *RISC*-архитектуры.

13 Назовите основные группы команд системы команд вычислительной машины.

14 В чем заключаются различия между Гарвардской и Принстонской архитектурами вычислительной машины?

15 Какие шины более низкого уровня входят в состав системной шины (магистрали)?

16 Какие форматы адресности команд существуют?

17 Что такое исполнительный адрес операнда?

18 Что такое способ адресации применительно к машинной команде?

19 Какие способы адресации существуют? В чем их основные отличия?

20 Как определяется исполнительный адрес операнда при косвенной адресации?

21 Как определяется исполнительный адрес операнда при относительной адресации?

22 Что входит в структуру простейшего процессора?

23 Какие бывают группы регистров процессора?

24 Какие операции выполняет арифметико-логическое устройство процессора?

25 Какие регистры флагов содержит арифметико-логическое устройство процессора?

26 Какие основные регистры и счетчики содержит блок управляющих регистров?

ПРАКТИЧЕСКАЯ ЧАСТЬ

ПРАКТИЧЕСКАЯ РАБОТА № 1. СИСТЕМЫ СЧИСЛЕНИЯ

Цель: изучить принципы позиционной системы счисления; научиться выполнять перевод чисел из одной системы счисления в другую.

Задание

1 Определить свой вариант задания и в соответствии с ним выполнить переводы из одних систем счисления в другие согласно индивидуальным заданиям (таблицы 1.1, 1.2):

– целое число *X*1: из десятичной в двоичную, из десятичной в восьмеричную, из десятичной в троичную;

– целое число *X*2: из двоичной в десятичную, из двоичной в шестнадцатеричную;

– целое число *X3* из системы с основанием *Q* в десятичную;

– дробное десятичное число *X*4: из десятичной в двоичную, из десятичной в восьмеричную, из десятичной в троичную;

– дробное двоичное число *X*5: из двоичной в десятичную, из двоичной в шестнадцатеричную;

– дробное восьмеричное число *X*6 из восьмеричной в шестнадцатеричную.

2 По результатам расчетов подготовить отчет, который должен содержать результаты и основные шаги при выполнении переводов из одной системы счисления в другую. Ответы на задания практической работы (для самопроверки) приведены в приложении Б.

3 Продемонстрировать и пояснить преподавателю алгоритм работы используемых схем перевода с подробным описанием последовательности действий (правил перевода).

Таблица 1.1 – Варианты исходных данных для практической работы № 1. Перевод целых чисел из одной позиционной системы счисления в другую

Вариант	***X*1**	***X*2**	***X*3**	***Q***
1	132	1101010001	41A	16
2	474	10000000111	1186	9
3	777	1001110011	D49	16
4	567	101001000	3434	5
5	222	1100000000	1726	8
6	980	1101011111	7BF2A	16
7	616	1100001001	3D2C	16
8	178	1100100101	3127	8
9	901	1000110110	5BF	16
10	333	100100000	5143	6
11	279	1111100110	11101	16
12	123	101111111	10172	8

Вариант	***X*1**	***X*2**	***X*3**	***Q***
13	456	100001010	3638	16
14	457	110111101	C78	16
15	294	1110010111	7701	9
16	109	1110011100	146	16
17	587	111101111	6456	7
18	999	1000010011	3777	16
19	351	10010000	3A3B4	16
20	699	111001010	1DC8	16
21	311	110011010	101001	8
22	911	1111111110	12001	3
23	381	11000001	1ED54	16
24	512	11001111	1D32	16
25	101	1001011001	10133	5
26	500	1010010	34567	8
27	307	1100110101	F77	16
28	954	100001000	15C	16
29	128	100011001	33101	6
30	493	1010101010	1001101	8
31	102	100100100	11AB	16
32	664	1110111	41534	8
33	700	10000100	4320F	16
34	765	111001110	14756	8
35	429	101010101	30201	4
36	235	111111110	56510B	16
37	813	1100011000	104135	6
38	248	101100111	1F1B	16
39	357	11011001	521401	7
40	496	10011001	100002	4

Таблица 1.2 – Варианты исходных данных для практической работы № 1. Перевод чисел с плавающей точкой из одной системы счисления в другую

Вариант	***X*4**	***X*5**	***X*6**
1	22,08	1001000111,10011	671,24
2	34,07	1001100,110011	413,41
3	11,89	11101111,101	1017,2
4	69,86	110100010,10101	112,04
5	53,75	1001011001,011	1347,17

Вариант	***X*4**	***X*5**	***X*6**
6	87,27	101001101,001001	1665,3
7	97,14	1100110010,1101	465,3
8	11,11	1001011,0101	704,6
9	90,45	11010110,00001	666,16
10	33,22	1000010011,00101	140,22
11	17,80	1011111111,01001	1600,14
12	79,41	10110101,10111	1053,2
13	18,05	1110101100,1011	1471,17
14	10,15	100101010,00011	452,63
15	13,17	110011110,1000011	1034,34
16	64,03	1010001001,11011	605,02
17	29,10	1000001101,01	247,1
18	23,45	1011110000,100101	1446,62
19	66,50	1000110001,1011	1762,7
20	53,35	1010011111,1101	1164,36
21	67,14	1011010101,1	615,72
22	99,01	1100111001,1001	1343,66
23	37,09	111001000,01	171,3
24	48,09	1000000101,01011	750,51
25	53,21	1101101000,01	1335,2
26	19,91	10011000,1101011	1234,2
27	60,50	1110001101,1001	1031,5
28	70,04	1000000110,00101	150,44
29	88,39	1110010100,1011001	236,63
30	72,34	1010111010,1110111	1636,24
31	45,41	1110011100,111	101,01
32	11,99	1001111010,010001	144,3
33	22,11	1000001111,0101	611,42
34	20,68	10110011,01	156,01
35	27,99	110011000,111001	10,054
36	39,61	101110001,01101	100,4
37	10,10	100000110,10101	7,002
38	88,88	110110101,1	12,07
39	12,94	1010101010,1010	77,04
40	56,32	10001000,10001	456,23

ПРАКТИЧЕСКАЯ РАБОТА № 2. АРИФМЕТИЧЕСКИЕ ОПЕРАЦИИ НАД ПОЛОЖИТЕЛЬНЫМИ ЧИСЛАМИ

Цель: изучить базовые принципы арифметических операций в позиционных системах счисления; научиться производить арифметические действия над числами, представленными в различных системах счисления.

Задание

1 Определить свой вариант задания и произвести необходимые арифметические действия в соответствующих системах счисления согласно индивидуальным заданиям (таблицы 2.1, 2.2):

– сложить числа *A*1 и *A*2 в двоичной системе счисления;

– перемножить числа *A*1 и *A*2 в двоичной системе счисления;

– вычесть из числа *A*3 число *A*4 (*A*3 – *A*4) в двоичной системе счисления;

– разделить число *A*3 на число *A*4 (*A*3 / *A*4) в двоичной системе счисления;

– сложить числа *A*5 и *A*6 в восьмеричной системе счисления;

– перемножить числа *A*5 и *A*6 в восьмеричной системе счисления;

– сложить числа *A*7 и *A*8 в шестнадцатеричной системе счисления;

– перемножить числа *A*7 и *A*8 в шестнадцатеричной системе счисления.

2 По результатам расчетов подготовить отчет, который должен содержать результаты арифметических действий и пошаговое их выполнение.

3 Выполнить проверку расчетов. Для этого необходимо перевести исходные числа и результат каждого действия в десятичную систему счисления, произвести заданные действия в десятичной системе счисления и сравнить полученные результаты. Таблицы сложения и умножения в различных системах счисления приведены в приложении А, а ответы на задания практической работы (для самопроверки) приведены в приложении Б.

4 Продемонстрировать и пояснить преподавателю порядок выполнения расчетов (арифметических действий).

Таблица 2.1 – Варианты заданий для практической работы № 2. Положительные числа для выполнения арифметических операций в двоичной системе счисления

Вариант	***A*1**	***A*2**	***A*3**	***A*4**
1	0,11110	0,00110	1101010001	1010
2	101,0101	10,1010	10000000111	1000
3	1101,0	1101,0	1001110011	111001
4	110,1110	111,0110	101001000	101
5	11001,110	1011,10	1100000000	1000
6	1111,110	1010,11	1101011111	111
7	0,1101	0,10011	11000010	1100
8	10101,0	10101,0	11001001	1010
9	11110,0	11011,0	1000110110	100
10	10111,1	11110,10	100100000	10011

Вариант	***A*1**	***A*2**	***A*3**	***A*4**
11	101011,0	111111,1	1111100110	1111
12	110101,1	1101,1	101111111	1011
13	111,110	111,110	100001010	101010
14	111,011	101,011	110111101	10101
15	10111,0	11011,0	1110010111	111
16	10011,0	11,110	1110011100	101
17	111,10	11,010	111101111	11110
18	0,10011	1111,0	100100100	100
19	10101,0	1101,1	10010000	1000
20	11011,0	111,110	111001010	101
21	11110,1	101,011	110011010	110
22	111111,1	11011,0	1111111110	1111
23	1101,0	11,110	11000001	11
24	110,1011	11,110	11001111	1100
25	101,1100	11,010	1001011001	1001
26	111,1101	1101,0	1010010	10
27	0,11011	111,0110	1100110101	11
28	10101,0	1011,10	100001000	1000
29	1011001,0	1010,110	100011001	1011
30	11011001,0	101010,0	1010101010	101010
31	10011001,0	111,0	100100100	111
32	1010010,0	11101,0	1110111	11101
33	1110101,0	101010,0	10000100	1000
34	1001000,0	1111,0	111001110	101
35	11000010,0	10001,0	101010101	101010
36	11001001,0	11110,0	111111110	1111
37	0,11110	100,0	1100011000	10001
38	1000,0	1000,0	101100111	101
39	100,010	110011,1	11011001	110
40	1010,110	10101,0	10011001	10011

Таблица 2.2 – Варианты заданий для практической работы № 2. Числа для выполнения операций в восьмеричной и шестнадцатеричной системах счисления

Вариант	***A*5**	***A*6**	***A*7**	***A*8**
1	345,2	7,3	41A	ABCDE
2	147,37	5,26	118	CA335
3	0,625	0,23	D4	BAE43

Вариант	***A5***	***A6***	***A7***	***A8***
4	0,657	0,425	341	17AE
5	0,724	4,165	1725	FA71
6	1475,7	123,4	7BF	5241
7	3777	61,5	3D2C	145C
8	37,05	7,77	3127	DC56
9	0,707	0,775	5BF	AEF
10	375,63	34,15	5143	24CF
11	377,17	11,11	111B	AC179
12	766,65	3,33	10172	CDE7
13	0,747	7,345	367	ABCD
14	5,375	7,16	C778	1569
15	7272,3	15,6	701	2016
16	671,24	5,02	148	2E31
17	413,41	47,1	6256	505ABC
18	117,2	4,62	3777	ADE3
19	112,04	162,7	3A3B4	AF
20	1347,17	4,36	1DC8	FFFD
21	1665,3	15,72	10101	FFFFF1
22	465,3	33,66	12001	AB113
23	704,6	11,3	1ED4	4567
24	666,16	7,51	1DE2	12345
25	140,22	33,2	1013	CA333
26	1600,14	14,2	34567	5DE
27	1053,2	31,5	FA77	1980
28	1471,17	15,44	156C	FAFE
29	452,63	23,63	33101	BABA
30	1034,34	16,24	10001	DDCC
31	15	101,01	11AB	ACDC
32	25,4	144,3	4134	2AB3
33	6,7	611,42	4632	1ABC
34	7,3	156,01	1456	5A7C
35	17,5	10,054	30101	33AED
36	321	100,4	5D65	3F0A
37	125	7,002	1045	AAA1
38	25,45	12,07	1F2B	1110F
39	1,33	77,04	532	FF0011
40	30	456,23	1002	1A2B3C

ПРАКТИЧЕСКАЯ РАБОТА № 3. ОПЕРАЦИИ НА АРИФМЕТИКО-ЛОГИЧЕСКОМ УСТРОЙСТВЕ (АРИФМЕТИКА С ЦЕЛЫМИ АЛГЕБРАИЧЕСКИМИ ЧИСЛАМИ)

Цель: изучить принципы кодирования чисел (способы представления числа), принципы арифметических и логических операций над числами в ЭВМ; научиться производить операции над числами в разных кодах; научиться производить логические операции с двоичными кодами.

Задание

1 Определить свой вариант задания и в соответствии с ним произвести необходимые расчеты (таблицы 3.1, 3.2):

– выполнить логическую операцию *A*1 над двоичным представлением чисел $X1_{(10)}$ и $X2_{(10)}$;

– выполнить логическую операцию сдвига *A*2 для целого беззнакового числа $X3_{(10)}$ (*использовать двоичное машинное слово*);

– выполнить операции арифметического сложения над целыми числами $X4_{(10)}$ и $X5_{(10)}$ в дополнительном и обратном двоичных кодах;

– выполнить операцию сдвига *A*3 над заданным модифицированным двоичным кодом $[X6]_{МК}$;

– вычислить результат применения последовательности операций (*Exp*) в арифметике чисел с фиксированной точкой над двоичными кодами чисел $X3_{(10)}$ и $X4_{(10)}$ (**внимание:** первыми всегда выполняются действия в скобках; вычисления выполнять в дополнительном коде).

2 Подготовить отчет, который должен содержать результаты расчетов и все выполненные до получения результата шаги.

При выполнении расчетов необходимо учесть, что входные числа и результат могут иметь заданную разрядность (*например, 16 бит*). Ответы на задания практической работы (для самопроверки) приведены в приложении Б.

3 Продемонстрировать и пояснить преподавателю порядок выполнения расчетов.

Таблица 3.1 – Исходные числа для практической работы № 3

Вариант	$X1_{(10)}$	$X2_{(10)}$	$X3_{(10)}$	$X4_{(10)}$	$X5_{(10)}$	$[X6]_{МК}$	**МК**
1	255	107	176	–55	65	10.01000110	МПК
2	250	255	134	–111	100	10.01000111	МОК
3	241	101	117	–77	99	10.01000111	МДК
4	216	43	211	–47	70	10.01000110	МПК
5	215	241	147	–89	54	00.00000111	МОК
6	211	123	56	–13	87	11.11010100	МДК
7	208	163	64	–128	97	00.01000110	МПК
8	205	77	47	–107	11	11.11111110	МОК

Вариант	$X1_{(10)}$	$X2_{(10)}$	$X3_{(10)}$	$X4_{(10)}$	$X5_{(10)}$	$[X6]_{МК}$	**МК**
9	202	175	66	–51	90	10.01110101	МДК
10	175	134	140	–109	33	00.01000110	МПК
11	163	111	160	–123	18	10.00000111	МОК
12	163	33	153	–1	79	10.11000111	МДК
13	156	145	141	–66	18	10.01010101	МПК
14	147	76	254	–101	11	00.00000110	МОК
15	145	156	234	–200	13	10.01000111	МДК
16	33	205	65	–39	64	11.01010101	МПК
17	139	147	247	–7	29	00.01000110	МОК
18	134	202	44	–69	69	11.01000111	МДК
19	133	163	251	–99	67	00.00110101	МПК
20	127	104	164	–100	54	00.00000110	МОК
21	124	216	156	–255	67	00.01000111	МДК
22	123	134	133	–222	99	10.01000111	МПК
23	114	211	171	–87	137	00.01010101	МОК
24	111	208	157	–50	48	11.11000110	МДК
25	107	108	151	–93	53	00.00010001	МПК
26	104	214	214	–45	20	11.11010000	МОК
27	101	32	130	–88	61	00.01000110	МДК
28	77	215	251	–6	70	11.01000111	МПК
29	76	124	236	–41	88	10.01110101	МОК
30	63	156	236	–16	72	00.11111111	МДК
31	43	101	101	–61	45	10.01011111	МПК

Таблица 3.2 – Варианты операций над числами для практической работы № 3

Вариант	***A1***	***A2***	***A3***	***Exp***
1	*AND*	*ROR* (3)	*SAR* (3)	$X3 – (X3 – X4)$
2	*AND*	*ROR* (4)	*SAR* (7)	$(X3 – X4) + X3$
3	*AND*	*ROR* (5)	*SAL* (3)	$(X3 – X4) – X3$
4	*OR*	*ROR* (6)	*SAL* (5)	$(X3 – X4) – X4$
5	*OR*	*ROR* (2)	*SAL* (4)	$(X4 – X3) \cdot (X3 – X4)$
6	*OR*	*ROR* (4)	*SAR* (2)	$(X4 – X3) + X3$
7	*XOR*	*ROR* (6)	*SAR* (6)	$(X3 – X4) – X4$
8	*XOR*	*ROR* (11)	*SAR* (10)	$X3 – (X4 + X4)$
9	*XOR*	*ROL* (3)	*SAL* (4)	$X3 + (NOT\ X4)$

Вариант	***A1***	***A2***	***A3***	***Exp***
10	*NOT*	*ROL* (5)	*SAL* (5)	(*X*4 – *X*3) · (*X*3 – *X*4)
11	*NOT*	*ROL* (7)	*SAL* (2)	*X*3 + (*X*4 – *X*3)
12	*NOT*	*ROL* (10)	*SAL* (3)	*X*3 – (*X*4 + *X*4)
13	*AND*	*ROL* (12)	*SAL* (4)	(*X*3 – *X*4) · (*X*3 – *X*4)
14	*AND*	*ROL* (8)	*SAR* (5)	(*X*3 – *X*4) – *X*4
15	*AND*	*ROL* (6)	*SAR* (8)	(*X*3 – *X*4) · (*X*3 – *X*4)
16	*OR*	*ROL* (4)	*SAR* (4)	(*X*3 + *X*3) – *X*4
17	*OR*	*SHR* (2)	*SAL* (2)	(*X*4 – *X*3) · (*X*3 – *X*4)
18	*OR*	*SHR* (1)	*SAL* (1)	(*X*4 – *X*3) – *X*3
19	*XOR*	*SHR* (6)	*SAL* (6)	(*X*4 – *X*3) – *X*3
20	*XOR*	*SHR* (3)	*SAL* (3)	(*X*3 – *X*4) · (*X*3 – *X*4)
21	*XOR*	*SHR* (4)	*SAL* (4)	*X*4 – (*X*3 + *X*3)
22	*NOT*	*SHR* (5)	*SAR* (5)	(*X*3 + *X*3) – *X*4
23	*NOT*	*SHR* (6)	*SAL* (2)	*X*4 – (*X*3 + *X*3)
24	*NOT*	*SHR* (7)	*SAL* (3)	(*X*4 – *X*3) + *X*3
25	*AND*	*SHL* (8)	*SAL* (5)	(*X*4 – *X*3) · (*X*4 – *X*3)
26	*AND*	*SHL* (4)	*SAL* (1)	*X*4 – (*X*3 + *X*3)
27	*AND*	*SHL* (7)	*SAR* (2)	*X*3 – (*X*4 + *X*4)
28	*OR*	*SHL* (2)	*SAR* (7)	(*X*4 – *X*3) – *X*3
29	*OR*	*SHL* (3)	*SAR* (8)	(*X*4 + *X*3) – (*X*4 – *X*3)
30	*OR*	*SHL* (5)	*SAR* (4)	(*X*3 + *X*3) – *X*4
31	*XOR*	*SHL* (1)	*SAR* (7)	(*X*4 + *X*3) – (*X*4 – *X*3)

Обозначения в таблице:

– *ROL* (*N*) – циклический сдвиг влево на *N* разрядов;

– *ROR* (*N*) – циклический сдвиг вправо на *N* разрядов;

– *SHL* (*N*) – логический сдвиг влево (*Shift Left*) на *N* разрядов;

– *SHR* (*N*) – логический сдвиг вправо (*Shift Right*) на *N* разрядов;

– *SAL* (*N*) – арифметический сдвиг влево (*Shift Arithmetic Left*) на *N* разрядов;

– *SAR* (*N*) – арифметический сдвиг вправо (*Shift Arithmetic Right*) на *N* разрядов;

– *XOR* – логическая операция «сумма по модулю 2».

ПРАКТИЧЕСКАЯ РАБОТА № 4. АРИФМЕТИКА ВЕЩЕСТВЕННЫХ ЧИСЕЛ, ПРЕДСТАВЛЕННЫХ В ФОРМАТЕ С ПЛАВАЮЩЕЙ ТОЧКОЙ (ЗАПЯТОЙ)

Цель: изучить принципы кодирования чисел (способы представления числа), принципы арифметических операций над вещественными числами в ЭВМ; научиться производить операции над числами в разных кодах; научиться производить операции над числами с плавающей точкой.

Задание

1 Определить свой вариант задания и в соответствии с ним произвести необходимые расчеты (таблица 4.1):

– представить вещественное число $X_{(10)}$ в формате одинарной точности (Y_{SINGLE}) в соответствии со стандартом *IEEE* 754–2019;

– найти сумму S вещественных чисел $A_{(2)}$ и $B_{(2)}$ в формате с плавающей точкой (запятой), если эти числа представлены в виде порядков A_E и B_E и мантисс A_M и B_M соответственно. При сложении использовать модифицированный дополнительный код, а результат представить в модифицированном прямом коде;

– найти произведение P вещественных чисел $A_{(2)}$ и $B_{(2)}$ в формате с плавающей точкой (запятой), если эти числа представлены в виде порядков A_E и B_E и мантисс A_M и B_M соответственно. При умножении использовать модифицированный обратный код, а результат представить в модифицированном прямом коде;

– найти частное Q вещественных чисел $C_{(2)}$ и $D_{(2)}$ в формате с плавающей точкой (запятой), если эти числа представлены в виде порядков C_E и D_E и мантисс C_M и D_M соответственно. Мантиссы чисел представляют собой правильную дробь (с нулевой целой частью). При делении использовать модифицированный обратный код, а результат представить в прямом коде в виде порядка Q_E и мантиссы Q_M.

2 Подготовить отчет, который должен содержать результаты расчетов и все выполненные до получения результата шаги. Ответы на задания практической работы (для самопроверки) приведены в приложении Б.

3 Продемонстрировать и пояснить преподавателю порядок выполнения расчетов.

Таблица 4.1 – Исходные числа для практической работы № 4

Вариант	$X_{(10)}$	$A_{E(2)}$	$B_{E(2)}$	$A_{M(2)}$	$B_{M(2)}$	$C_{E(2)}$	$D_{E(2)}$	$C_{M(2)}$	$D_{M(2)}$
1	+168,748	1.001	0.010	0.111111	1.001000	1.010	0.000	0.1111	0.1000
2	–326,115	0.010	1.010	0.111000	0.101110	1.010	0.010	1.1110	0.1001
3	–232,753	0.011	0.001	1.000110	0.110001	0.100	0.010	0.1101	1.1010
4	+196,455	1.001	1.010	0.010101	0.101010	1.010	0.000	1.1100	1.1011
5	–132,785	0.010	1.001	1.100001	1.101101	0.010	1.010	0.1001	0.1100
6	+187,625	0.100	0.010	1.001100	0.101110	1.010	0.010	1.1000	0.1101
7	+149,753	1.011	0.001	1.111000	0.100001	0.011	1.001	0.0111	1.1110
8	–341,781	0.010	1.001	0.001111	0.110010	0.000	0.011	1.0110	1.1111

Вариант	$X_{(10)}$	$A_{E(2)}$	$B_{E(2)}$	$A_{M(2)}$	$B_{M(2)}$	$C_{E(2)}$	$D_{E(2)}$	$C_{M(2)}$	$D_{M(2)}$
9	+245,246	0.011	0.000	1.000100	1.110001	1.010	0.001	0.0101	0.0111
10	+214,321	1.010	0.010	0.100010	1.100010	0.100	1.010	1.0100	0.0110
11	–244,789	0.100	1.001	1.100100	0.011011	0.000	1.010	0.0011	1.0101
12	–175,852	0.011	0.001	1.010010	1.011110	0.010	1.010	1.0010	1.0100
13	–162,364	1.010	0.010	1.101101	1.100000	0.010	0.100	0.0001	0.0011
14	+303,159	1.011	0.000	0.001110	0.001111	0.000	1.010	1.1010	0.1011
15	–212,539	0.001	1.100	0.000101	1.111011	1.010	0.010	0.1011	1.1110
16	–222,725	0.010	1.001	1.001000	0.010101	0.010	1.010	0.1000	0.1111
17	+141,125	1.010	0.010	0.101110	1.100001	1.001	0.011	0.1001	1.1110
18	–158,562	0.001	0.011	0.110001	1.001100	0.011	0.000	1.1010	0.1101
19	–190,375	1.010	1.001	0.101010	1.111000	0.001	1.010	1.1011	1.1100
20	+177,825	1.001	0.010	1.101101	0.001111	1.010	0.100	0.1100	0.1001
21	+283,763	0.010	0.100	0.101110	1.000100	0.010	0.100	0.1101	1.1000
22	–309,750	0.001	1.011	0.100001	0.100010	1.010	1.010	1.1110	0.0111
23	+305,500	1.001	0.010	0.110010	1.100100	1.010	1.010	1.1111	1.0110
24	–180,458	0.000	0.011	1.110001	1.010010	0.100	0.100	0.0111	0.0101
25	+316,625	0.010	1.010	1.100010	1.101101	1.010	1.010	0.0110	1.0100
26	+140,329	1.001	0.100	0.011011	0.001110	0.010	0.010	1.0101	0.0011
27	–166,450	0.001	0.011	1.011110	0.000101	1.010	1.010	1.0100	1.0010
28	+327,723	0.010	1.010	1.100000	0.111111	0.011	0.011	0.0011	0.0001
29	–114,352	0.000	1.011	0.001111	0.111000	0.000	0.000	0.1011	1.1010
30	–133,750	1.100	0.001	1.111011	1.000110	1.010	1.010	1.1110	0.1011
31	+277,150	0.011	1.011	0.101010	1.000100	0.100	0.100	0.0110	0.1010

КОНТРОЛЬНАЯ РАБОТА. ПРЕДСТАВЛЕНИЕ И ОБРАБОТКА ИНФОРМАЦИИ В ЭВМ

Задание

1 Индивидуальный вариант задания контрольной работы (КР) соответствует порядковому номеру в списке группы. Значение *X* соответствует первой цифре номера группы студента и необходимо для определения варианта задания из практических работ (ПР) № 1–4 по таблице К.1. Например, студент 7-й в списке группы 481571 ($X = 4$), тогда он выполняет варианты: 29-й (4 + 25) из ПР № 1, 31-й (4 + 27) из ПР № 2, 9-й (4 + 5) из ПР № 3, 11-й (4 + 7) из ПР № 4.

2 Для установленного варианта КР выполнить необходимые расчеты из указанных в таблице К.1 практических работ (с. 101, 104, 107, 110) в соответствии с определенными вариантами их заданий, знать ответы на контрольные вопросы (с. 52, 78, 98), уметь объяснить каждый шаг процесса решения своего варианта индивидуального задания КР.

Таблица К.1 – Варианты индивидуальных заданий контрольной работы

Вариант задания КР	**Номер варианта задания из практической работы**			
	ПР № 1 (с. 101)	**ПР № 2 (с. 104)**	**ПР № 3 (с. 107)**	**ПР № 4 (с. 110)**
1	*X* + 31	*X* + 21	*X* + 11	*X* + 1
2	*X* + 30	*X* + 22	*X* + 10	*X* + 2
3	*X* + 29	*X* + 23	*X* + 9	*X* + 3
4	*X* + 28	*X* + 24	*X* + 8	*X* + 4
5	*X* + 27	*X* + 25	*X* + 7	*X* + 5
6	*X* + 26	*X* + 26	*X* + 6	*X* + 6
7	*X* + 25	*X* + 27	*X* + 5	*X* + 7
8	*X* + 24	*X* + 28	*X* + 4	*X* + 8
9	*X* + 23	*X* + 29	*X* + 3	*X* + 9
10	*X* + 22	*X* + 30	*X* + 2	*X* + 3
11	*X* + 21	*X* + 31	*X* + 1	*X* + 5
12	*X* + 20	*X* + 1	*X* + 0	*X* + 7
13	*X* + 19	*X* + 2	*X* + 12	*X* + 9
14	*X* + 18	*X* + 3	*X* + 13	*X* + 11
15	*X* + 17	*X* + 4	*X* + 14	*X* + 13
16	*X* + 16	*X* + 5	*X* + 15	*X* + 15
17	*X* + 15	*X* + 6	*X* + 16	*X* + 17
18	*X* + 14	*X* + 7	*X* + 17	*X* + 19
19	*X* + 13	*X* + 8	*X* + 18	*X* + 21
20	*X* + 12	*X* + 9	*X* + 19	*X* + 0
21	*X* + 11	*X* + 10	*X* + 20	*X* + 1
22	*X* + 10	*X* + 11	*X* + 21	*X* + 2
23	*X* + 9	*X* + 12	*X* + 22	*X* + 4
24	*X* + 8	*X* + 13	*X* + 3	*X* + 6
25	*X* + 7	*X* + 14	*X* + 5	*X* + 8
26	*X* + 6	*X* + 15	*X* + 7	*X* + 10
27	*X* + 5	*X* + 16	*X* + 9	*X* + 12
28	*X* + 4	*X* + 17	*X* + 11	*X* + 14
29	*X* + 3	*X* + 18	*X* + 13	*X* + 16
30	*X* + 2	*X* + 19	*X* + 15	*X* + 18
31	*X* + 1	*X* + 20	*X* + 17	*X* + 20
32	*X* + 0	*X* + 21	*X* + 19	*X* + 22

ПРИЛОЖЕНИЕ А

Расчетные таблицы арифметических операций в различных системах счисления, используемых в компьютере

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

×	**0**	**1**
0	0	0
1	0	1

Рисунок А.1 – Таблицы для выполнения арифметических действий (сложение и умножение) в двоичной системе счисления

+	**0**	**1**	**2**
0	0	1	2
1	1	2	10
2	2	10	11

×	**0**	**1**	**2**
0	0	0	0
1	0	1	2
2	0	2	11

Рисунок А.2 – Таблицы для выполнения арифметических действий (сложение и умножение) в троичной системе счисления

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

×	**1**	**2**	**3**	**4**	**5**	**6**	**7**
1	1	2	3	4	5	6	7
2	2	4	6	10	12	14	16
3	3	6	11	14	17	22	25
4	4	10	14	20	24	30	34
5	5	12	17	24	31	36	43
6	6	14	22	30	36	44	52
7	7	16	25	34	43	52	61

Рисунок А.3 – Таблицы для выполнения арифметических действий (сложение и умножение) в восьмеричной системе счисления

+	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10
2	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11
3	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12
4	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13
5	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14
6	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15
7	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16
8	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17
9	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18
A	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19
B	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A
C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B
D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C
E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D
F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E

Рисунок А.4 – Таблица для выполнения арифметического действия сложения в шестнадцатеричной системе счисления

×	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2	2	4	6	8	A	C	E	10	12	14	16	18	1A	1C	1E
3	3	6	9	C	F	12	15	18	1B	1E	21	24	27	2A	2D
4	4	8	C	10	14	18	1C	20	24	28	2C	30	34	38	3C
5	5	A	F	14	19	1E	23	28	2D	32	37	3C	41	46	4B
6	6	C	12	18	1E	24	2A	30	36	3C	42	48	4E	54	5A
7	7	E	15	1C	23	2A	31	38	3F	46	4D	54	5B	62	69
8	8	10	18	20	28	30	38	40	48	50	58	60	68	70	78
9	9	12	1B	24	2D	36	3F	48	51	5A	63	6C	75	7E	87
A	A	14	1E	28	32	3C	46	50	5A	64	6E	78	82	8C	96
B	B	16	21	2C	37	42	4D	58	63	6E	79	84	8F	9A	A5
C	C	18	24	30	3C	48	54	60	6C	78	84	90	9C	A8	B4
D	D	1A	27	34	41	4E	5B	68	75	82	8F	9C	A9	B6	C3
E	E	1C	2A	38	46	54	62	70	7E	8C	9A	A8	B6	C4	D2
F	F	1E	2D	3C	4B	5A	69	78	87	96	A5	B4	C3	D2	E1

Рисунок А.5 – Таблицы для выполнения арифметического действия умножения в шестнадцатеричной системе счисления

ПРИЛОЖЕНИЕ Б

Ответы на практические задания (для самопроверки)

Таблица Б.1 – Ответы к практической работе № 1. Переводы чисел *X*1 и *X*4

Вариант	*X*1	*X*4
1	$X1_{(2)} = 10000100$	$X4_{(2)} = 10110{,}00010100011$
	$X1_{(8)} = 204$	$X4_{(8)} = 26{,}05075341217$
	$X1_{(3)} = 11220$	$X4_{(3)} = 211{,}00201102212$
2	$X1_{(2)} = 111011010$	$X4_{(2)} = 100010{,}00010001111$
	$X1_{(8)} = 732$	$X4_{(8)} = 42{,}04365605075$
	$X1_{(3)} = 122120$	$X4_{(3)} = 1021{,}00122000021$
3	$X1_{(2)} = 1100001001$	$X4_{(2)} = 1011{,}11100011110$
	$X1_{(8)} = 1411$	$X4_{(8)} = 13{,}70753412172$
	$X1_{(3)} = 1001210$	$X4_{(3)} = 102{,}22000021021$
4	$X1_{(2)} = 1000110111$	$X4_{(2)} = 1000101{,}11011100001$
	$X1_{(8)} = 1067$	$X4_{(8)} = 105{,}67024365605$
	$X1_{(3)} = 210000$	$X4_{(3)} = 2120{,}21201222110$
5	$X1_{(2)} = 11011110$	$X4_{(2)} = 110101{,}11$
	$X1_{(8)} = 336$	$X4_{(8)} = 65{,}6$
	$X1_{(3)} = 22020$	$X4_{(3)} = 1222{,}20202020202$
6	$X1_{(2)} = 1111010100$	$X4_{(2)} = 1010111{,}01000101000$
	$X1_{(8)} = 1724$	$X4_{(8)} = 127{,}21217270243$
	$X1_{(3)} = 1100022$	$X4_{(3)} = 10020{,}02102121110$
7	$X1_{(2)} = 1001101000$	$X4_{(2)} = 1100001{,}00100011110$
	$X1_{(8)} = 1150$	$X4_{(8)} = 141{,}10753412172$
	$X1_{(3)} = 211211$	$X4_{(3)} = 10121{,}01021000112$
8	$X1_{(2)} = 10110010$	$X4_{(2)} = 1011{,}00011100001$
	$X1_{(8)} = 262$	$X4_{(8)} = 13{,}07024365605$
	$X1_{(3)} = 20121$	$X4_{(3)} = 102{,}00222201201$
9	$X1_{(2)} = 1110000101$	$X4_{(2)} = 1011010{,}01110011001$
	$X1_{(8)} = 1605$	$X4_{(8)} = 132{,}34631463146$
	$X1_{(3)} = 1020101$	$X4_{(3)} = 10100{,}11001100110$
10	$X1_{(2)} = 101001101$	$X4_{(2)} = 100001{,}00111000010$
	$X1_{(8)} = 515$	$X4_{(8)} = 41{,}16050753412$
	$X1_{(3)} = 110100$	$X4_{(3)} = 1020{,}01222110102$
11	$X1_{(2)} = 100010111$	$X4_{(2)} = 10001{,}11001100110$
	$X1_{(8)} = 427$	$X4_{(8)} = 21{,}63146314631$
	$X1_{(3)} = 101100$	$X4_{(3)} = 122{,}21012101210$
12	$X1_{(2)} = 1111011$	$X4_{(2)} = 1001111{,}01101000111$
	$X1_{(8)} = 173$	$X4_{(8)} = 117{,}32172702436$
	$X1_{(3)} = 11120$	$X4_{(3)} = 2221{,}10200122000$

Вариант	*X*1	*X*4
13	$X1_{(2)}$ = 111001000	$X4_{(2)}$ = 10010,00001100110
	$X1_{(8)}$ = 710	$X4_{(8)}$ = 22,03146314631
	$X1_{(3)}$ = 121220	$X4_{(3)}$ = 200,00110011001
14	$X1_{(2)}$ = 111001001	$X4_{(2)}$ = 1010,00100110011
	$X1_{(8)}$ = 711	$X4_{(8)}$ = 12,11463146314
	$X1_{(3)}$ = 121221	$X4_{(3)}$ = 101,01100110011
15	$X1_{(2)}$ = 100100110	$X4_{(2)}$ = 1101,00101011100
	$X1_{(8)}$ = 446	$X4_{(8)}$ = 15,12702436560
	$X1_{(3)}$ = 101220	$X4_{(3)}$ = 111,01112022100
16	$X1_{(2)}$ = 1101101	$X4_{(2)}$ = 1000000,00000111101
	$X1_{(8)}$ = 155	$X4_{(8)}$ = 100,01727024365
	$X1_{(3)}$ = 11001	$X4_{(3)}$ = 2101,00021021211
17	$X1_{(2)}$ = 1001001011	$X4_{(2)}$ = 11101,00011001100
	$X1_{(8)}$ = 1113	$X4_{(8)}$ = 35,06314631463
	$X1_{(3)}$ = 210202	$X4_{(3)}$ = 1002,00220022002
18	$X1_{(2)}$ = 1111100111	$X4_{(2)}$ = 10111,01110011001
	$X1_{(8)}$ = 1747	$X4_{(8)}$ = 27,34631463146
	$X1_{(3)}$ = 1101000	$X4_{(3)}$ = 212,11001100110
19	$X1_{(2)}$ = 101011111	$X4_{(2)}$ = 1000010,1
	$X1_{(8)}$ = 537	$X4_{(8)}$ = 102,4
	$X1_{(3)}$ = 111000	$X4_{(3)}$ = 2110,11111111111
20	$X1_{(2)}$ = 1010111011	$X4_{(2)}$ = 110101,01011001100
	$X1_{(8)}$ = 1273	$X4_{(8)}$ = 65,26314631463
	$X1_{(3)}$ = 221220	$X4_{(3)}$ = 1222,10011001100
21	$X1_{(2)}$ = 100110111	$X4_{(2)}$ = 1000011,00100011110
	$X1_{(8)}$ = 467	$X4_{(8)}$ = 103,10753412172
	$X1_{(3)}$ = 102112	$X4_{(3)}$ = 2111,01021000112
22	$X1_{(2)}$ = 1110001111	$X4_{(2)}$ = 1100011,00000010100
	$X1_{(8)}$ = 1617	$X4_{(8)}$ = 143,00507534121
	$X1_{(3)}$ = 1020202	$X4_{(3)}$ = 10200,00002102121
23	$X1_{(2)}$ = 101111101	$X4_{(2)}$ = 100101,00010111000
	$X1_{(8)}$ = 575	$X4_{(8)}$ = 45,05605075341
	$X1_{(3)}$ = 112010	$X4_{(3)}$ = 1101,00210212111
24	$X1_{(2)}$ = 1000000000	$X4_{(2)}$ = 110000,00010111000
	$X1_{(8)}$ = 1000	$X4_{(8)}$ = 60,05605075341
	$X1_{(3)}$ = 200222	$X4_{(3)}$ = 1210,00210212111
25	$X1_{(2)}$ = 1100101	$X4_{(2)}$ = 110101,00110101110
	$X1_{(8)}$ = 145	$X4_{(8)}$ = 65,15341217270
	$X1_{(3)}$ = 10202	$X4_{(3)}$ = 1222,01220000210

Вариант	*X*1	*X*4
26	$X1_{(2)}$ = 111110100	$X4_{(2)}$ = 10011,11101000111
	$X1_{(8)}$ = 764	$X4_{(8)}$ = 23,72172702436
	$X1_{(3)}$ = 200112	$X4_{(3)}$ = 201,22012010111
27	$X1_{(2)}$ = 100110011	$X4_{(2)}$ = 111100,1
	$X1_{(8)}$ = 463	$X4_{(8)}$ = 74,4
	$X1_{(3)}$ = 102101	$X4_{(3)}$ = 2020,11111111111
28	$X1_{(2)}$ = 1110111010	$X4_{(2)}$ = 1000110,00001010001
	$X1_{(8)}$ = 1672	$X4_{(8)}$ = 106,02436560507
	$X1_{(3)}$ = 1022100	$X4_{(3)}$ = 2121,00100201102
29	$X1_{(2)}$ = 10000000	$X4_{(2)}$ = 1011000,01100011110
	$X1_{(8)}$ = 200	$X4_{(8)}$ = 130,30753412172
	$X1_{(3)}$ = 11202	$X4_{(3)}$ = 10021,10111202210
30	$X1_{(2)}$ = 111101101	$X4_{(2)}$ = 1001000,01010111000
	$X1_{(8)}$ = 755	$X4_{(8)}$ = 110,25605075341
	$X1_{(3)}$ = 200021	$X4_{(3)}$ = 2200,10001121201
31	$X1_{(2)}$ = 1100110	$X4_{(2)}$ = 101101,01101000111
	$X1_{(8)}$ = 146	$X4_{(8)}$ = 55,32172702436
	$X1_{(3)}$ = 10210	$X4_{(3)}$ = 1200,10200122000
32	$X1_{(2)}$ = 1010011000	$X4_{(2)}$ = 1011,11111101011
	$X1_{(8)}$ = 1230	$X4_{(8)}$ = 13,77270243656
	$X1_{(3)}$ = 220121	$X4_{(3)}$ = 102,22220120101
33	$X1_{(2)}$ = 1010111100	$X4_{(2)}$ = 10110,00011100001
	$X1_{(8)}$ = 1274	$X4_{(8)}$ = 26,07024365605
	$X1_{(3)}$ = 221221	$X4_{(3)}$ = 211,00222201201
34	$X1_{(2)}$ = 1011111101	$X4_{(2)}$ = 10100,10101110000
	$X1_{(8)}$ = 1375	$X4_{(8)}$ = 24,53412172702
	$X1_{(3)}$ = 1001100	$X4_{(3)}$ = 202,20010020110
35	$X1_{(2)}$ = 110101101	$X4_{(2)}$ = 11011,11111101011
	$X1_{(8)}$ = 655	$X4_{(8)}$ = 33,77270243656
	$X1_{(3)}$ = 120220	$X4_{(3)}$ = 1000,22220120101
36	$X1_{(2)}$ = 11101011	$X4_{(2)}$ = 100111,10011100001
	$X1_{(8)}$ = 353	$X4_{(8)}$ = 47,47024365605
	$X1_{(3)}$ = 22201	$X4_{(3)}$ = 1110,12111020012
37	$X1_{(2)}$ = 1100101101	$X4_{(2)}$ = 1010,00011001100
	$X1_{(8)}$ = 1455	$X4_{(8)}$ = 12,06314631463
	$X1_{(3)}$ = 1010010	$X4_{(3)}$ = 101,00220022002
38	$X1_{(2)}$ = 11111000	$X4_{(2)}$ = 1011000,11100001010
	$X1_{(8)}$ = 370	$X4_{(8)}$ = 130,70243656050
	$X1_{(3)}$ = 100012	$X4_{(3)}$ = 10021,21220211200

Вариант	*X*1	*X*4
39	$X1_{(2)}$ = 101100101	$X4_{(2)}$ = 1100,11110000101
	$X1_{(8)}$ = 545	$X4_{(8)}$ = 14,74121727024
	$X1_{(3)}$ = 111020	$X4_{(3)}$ = 110,22110102100
40	$X1_{(2)}$ = 111110000	$X4_{(2)}$ = 111000,01010001111
	$X1_{(8)}$ = 760	$X4_{(8)}$ = 70,24365605075
	$X1_{(3)}$ = 200101	$X4_{(3)}$ = 2002,02212202112

Таблица Б.2 – Ответы к практической работе № 1. Переводы чисел *X*2 и *X*3

Вариант	*X*2		*X*3
1	$X2_{(10)}$ = 849	$X2_{(16)}$ = 351	$X3_{(10)}$ = 1050
2	$X2_{(10)}$ = 1031	$X2_{(16)}$ = 407	$X3_{(10)}$ = 888
3	$X2_{(10)}$ = 627	$X2_{(16)}$ = 273	$X3_{(10)}$ = 3401
4	$X2_{(10)}$ = 328	$X2_{(16)}$ = 148	$X3_{(10)}$ = 494
5	$X2_{(10)}$ = 768	$X2_{(16)}$ = 300	$X3_{(10)}$ = 982
6	$X2_{(10)}$ = 863	$X2_{(16)}$ = 35F	$X3_{(10)}$ = 507690
7	$X2_{(10)}$ = 777	$X2_{(16)}$ = 309	$X3_{(10)}$ = 15660
8	$X2_{(10)}$ = 805	$X2_{(16)}$ = 325	$X3_{(10)}$ = 1623
9	$X2_{(10)}$ = 566	$X2_{(16)}$ = 236	$X3_{(10)}$ = 1471
10	$X2_{(10)}$ = 288	$X2_{(16)}$ = 120	$X3_{(10)}$ = 1143
11	$X2_{(10)}$ = 998	$X2_{(16)}$ = 3E6	$X3_{(10)}$ = 69889
12	$X2_{(10)}$ = 383	$X2_{(16)}$ = 17F	$X3_{(10)}$ = 4218
13	$X2_{(10)}$ = 266	$X2_{(16)}$ = 10A	$X3_{(10)}$ = 13880
14	$X2_{(10)}$ = 445	$X2_{(16)}$ = 1BD	$X3_{(10)}$ = 3192
15	$X2_{(10)}$ = 919	$X2_{(16)}$ = 397	$X3_{(10)}$ = 5671
16	$X2_{(10)}$ = 924	$X2_{(16)}$ = 39C	$X3_{(10)}$ = 326
17	$X2_{(10)}$ = 495	$X2_{(16)}$ = 1EF	$X3_{(10)}$ = 2295
18	$X2_{(10)}$ = 531	$X2_{(16)}$ = 213	$X3_{(10)}$ = 14199
19	$X2_{(10)}$ = 144	$X2_{(16)}$ = 90	$X3_{(10)}$ = 238516
20	$X2_{(10)}$ = 458	$X2_{(16)}$ = 1CA	$X3_{(10)}$ = 7624
21	$X2_{(10)}$ = 410	$X2_{(16)}$ = 19A	$X3_{(10)}$ = 33281
22	$X2_{(10)}$ = 1022	$X2_{(16)}$ = 3FE	$X3_{(10)}$ = 136
23	$X2_{(10)}$ = 193	$X2_{(16)}$ = C1	$X3_{(10)}$ = 126292
24	$X2_{(10)}$ = 207	$X2_{(16)}$ = CF	$X3_{(10)}$ = 7474
25	$X2_{(10)}$ = 601	$X2_{(16)}$ = 259	$X3_{(10)}$ = 668
26	$X2_{(10)}$ = 82	$X2_{(16)}$ = 52	$X3_{(10)}$ = 14711
27	$X2_{(10)}$ = 821	$X2_{(16)}$ = 335	$X3_{(10)}$ = 3959
28	$X2_{(10)}$ = 264	$X2_{(16)}$ = 108	$X3_{(10)}$ = 348
29	$X2_{(10)}$ = 281	$X2_{(16)}$ = 119	$X3_{(10)}$ = 4573
30	$X2_{(10)}$ = 682	$X2_{(16)}$ = 2AA	$X3_{(10)}$ = 262721

Вариант	***X*2**		***X*3**
31	$X2_{(10)}$ = 292	$X2_{(16)}$ = 124	$X3_{(10)}$ = 4523
32	$X2_{(10)}$ = 119	$X2_{(16)}$ = 77	$X3_{(10)}$ = 17244
33	$X2_{(10)}$ = 132	$X2_{(16)}$ = 84	$X3_{(10)}$ = 274959
34	$X2_{(10)}$ = 462	$X2_{(16)}$ = 1CE	$X3_{(10)}$ = 6638
35	$X2_{(10)}$ = 341	$X2_{(16)}$ = 155	$X3_{(10)}$ = 801
36	$X2_{(10)}$ = 510	$X2_{(16)}$ = 1FE	$X3_{(10)}$ = 5656843
37	$X2_{(10)}$ = 792	$X2_{(16)}$ = 318	$X3_{(10)}$ = 8699
38	$X2_{(10)}$ = 359	$X2_{(16)}$ = 167	$X3_{(10)}$ = 7963
39	$X2_{(10)}$ = 217	$X2_{(16)}$ = D9	$X3_{(10)}$ = 89377
40	$X2_{(10)}$ = 153	$X2_{(16)}$ = 99	$X3_{(10)}$ = 1026

Таблица Б.3 – Ответы к практической работе № 1. Переводы чисел *X*5 и *X*6

Вариант	***X*5**	***X*6**
1	$X5_{(10)}$ = 583,59375	$X6_{(2)}$ = 110111001,0101
	$X5_{(16)}$ = 247,98	$X6_{(16)}$ = 1B9,5
2	$X5_{(10)}$ = 76,796875	$X6_{(2)}$ = 100001011,100001
	$X5_{(16)}$ = 4C,CC	$X6_{(16)}$ = 10B,84
3	$X5_{(10)}$ = 239,625	$X6_{(2)}$ = 1000001111,01
	$X5_{(16)}$ = EF,A	$X6_{(16)}$ = 20F,4
4	$X5_{(10)}$ = 418,65625	$X6_{(2)}$ = 1001010,0001
	$X5_{(16)}$ = 1A2,A8	$X6_{(16)}$ = 4A,1
5	$X5_{(10)}$ = 601,375	$X6_{(2)}$ = 1011100111,001111
	$X5_{(16)}$ = 259,6	$X6_{(16)}$ = 2E7,3C
6	$X5_{(10)}$ = 333,140625	$X6_{(2)}$ = 1110110101,011
	$X5_{(16)}$ = 14D,24	$X6_{(16)}$ = 3B5,6
7	$X5_{(10)}$ = 818,8125	$X6_{(2)}$ = 100110101,011
	$X5_{(16)}$ = 332,D	$X6_{(16)}$ = 135,6
8	$X5_{(10)}$ = 75,3125	$X6_{(2)}$ = 111000100,11
	$X5_{(16)}$ = 4B,5	$X6_{(16)}$ = 1C4,C
9	$X5_{(10)}$ = 214,03125	$X6_{(2)}$ = 110110110,00111
	$X5_{(16)}$ = D6,08	$X6_{(16)}$ = 1B6,38
10	$X5_{(10)}$ = 531,15625	$X6_{(2)}$ = 1100000,01001
	$X5_{(16)}$ = 213,28	$X6_{(16)}$ = 60,48
11	$X5_{(10)}$ = 767,28125	$X6_{(2)}$ = 1110000000,0011
	$X5_{(16)}$ = 2FF,48	$X6_{(16)}$ = 380,3
12	$X5_{(10)}$ = 181,71875	$X6_{(2)}$ = 1000101011,01
	$X5_{(16)}$ = B5,B8	$X6_{(16)}$ = 22B,4
13	$X5_{(10)}$ = 940,6875	$X6_{(2)}$ = 1100111001,001111
	$X5_{(16)}$ = 3AC,B	$X6_{(16)}$ = 339,3C

Вариант	*X*5	*X*6
14	$X5_{(10)}$ = 298,09375	$X6_{(2)}$ = 100101010,110011
	$X5_{(16)}$ = 12A,18	$X6_{(16)}$ = 12A,CC
15	$X5_{(10)}$ = 414,5234375	$X6_{(2)}$ = 1000011100,0111
	$X5_{(16)}$ = 19E,86	$X6_{(16)}$ = 21C,7
16	$X5_{(10)}$ = 649,84375	$X6_{(2)}$ = 110000101,00001
	$X5_{(16)}$ = 289,D8	$X6_{(16)}$ = 185,08
17	$X5_{(10)}$ = 525,25	$X6_{(2)}$ = 10100111,001
	$X5_{(16)}$ = 20D,4	$X6_{(16)}$ = A7,2
18	$X5_{(10)}$ = 752,578125	$X6_{(2)}$ = 1100100110,11001
	$X5_{(16)}$ = 2F0,94	$X6_{(16)}$ = 326,C8
19	$X5_{(10)}$ = 561,6875	$X6_{(2)}$ = 1111110010,111
	$X5_{(16)}$ = 231,B	$X6_{(16)}$ = 3F2,E
20	$X5_{(10)}$ = 671,8125	$X6_{(2)}$ = 1001110100,01111
	$X5_{(16)}$ = 29F,D	$X6_{(16)}$ = 274,78
21	$X5_{(10)}$ = 725,5	$X6_{(2)}$ = 110001101,11101
	$X5_{(16)}$ = 2D5,8	$X6_{(16)}$ = 18D,E8
22	$X5_{(10)}$ = 825,5625	$X6_{(2)}$ = 1011100011,11011
	$X5_{(16)}$ = 339,9	$X6_{(16)}$ = 2E3,D8
23	$X5_{(10)}$ = 456,25	$X6_{(2)}$ = 1111001,011
	$X5_{(16)}$ = 1C8,4	$X6_{(16)}$ = 79,6
24	$X5_{(10)}$ = 517,34375	$X6_{(2)}$ = 111101000,101001
	$X5_{(16)}$ = 205,58	$X6_{(16)}$ = 1E8,A4
25	$X5_{(10)}$ = 872,25	$X6_{(2)}$ = 1011011101,01
	$X5_{(16)}$ = 368,4	$X6_{(16)}$ = 2DD,4
26	$X5_{(10)}$ = 152,8359375	$X6_{(2)}$ = 1010011100,01
	$X5_{(16)}$ = 98,D6	$X6_{(16)}$ = 29C,4
27	$X5_{(10)}$ = 909,5625	$X6_{(2)}$ = 1000011001,101
	$X5_{(16)}$ = 38D,9	$X6_{(16)}$ = 219,A
28	$X5_{(10)}$ = 518,15625	$X6_{(2)}$ = 1101000,1001
	$X5_{(16)}$ = 206,28	$X6_{(16)}$ = 68,9
29	$X5_{(10)}$ = 916,6953125	$X6_{(2)}$ = 10011110,110011
	$X5_{(16)}$ = 394,B2	$X6_{(16)}$ = 9E,CC
30	$X5_{(10)}$ = 698,9296875	$X6_{(2)}$ = 1110011110,0101
	$X5_{(16)}$ = 2BA,EE	$X6_{(16)}$ = 39E,5
31	$X5_{(10)}$ = 924,875	$X6_{(2)}$ = 1000001,000001
	$X5_{(16)}$ = 39C,E	$X6_{(16)}$ = 41,04
32	$X5_{(10)}$ = 634,265625	$X6_{(2)}$ = 1100100,011
	$X5_{(16)}$ = 27A,44	$X6_{(16)}$ = 64,6

Вариант	*X*5	*X*6
33	$X5_{(10)}$ = 527,3125	$X6_{(2)}$ = 110001001,10001
	$X5_{(16)}$ = 20F,5	$X6_{(16)}$ = 189,88
34	$X5_{(10)}$ = 179,25	$X6_{(2)}$ = 1101110,000001
	$X5_{(16)}$ = B3,4	$X6_{(16)}$ = 6E,04
35	$X5_{(10)}$ = 408,890625	$X6_{(2)}$ = 1000,0001011
	$X5_{(16)}$ = 198,E4	$X6_{(16)}$ = 8,16
36	$X5_{(10)}$ = 369,40625	$X6_{(2)}$ = 1000000,1
	$X5_{(16)}$ = 171,68	$X6_{(16)}$ = 40,8
37	$X5_{(10)}$ = 262,65625	$X6_{(2)}$ = 111,00000001
	$X5_{(16)}$ = 106,A8	$X6_{(16)}$ = 7,01
38	$X5_{(10)}$ = 437,5	$X6_{(2)}$ = 1010,000111
	$X5_{(16)}$ = 1B5,8	$X6_{(16)}$ = A,1C
39	$X5_{(10)}$ = 682,625	$X6_{(2)}$ = 111111,0001
	$X5_{(16)}$ = 2AA,A	$X6_{(16)}$ = 3F,1
40	$X5_{(10)}$ = 136,53125	$X6_{(2)}$ = 100101110,010011
	$X5_{(16)}$ = 88,88	$X6_{(16)}$ = 12E,4C

Таблица Б.4 – Ответы к практической работе № 2. Результаты операций сложения и умножения чисел в двоичной системе счисления

Вариант	$(A1 + A2)_{(2)}$	$(A1 \times A2)_{(2)}$
1	1,001	0,0010110100
2	111,1111	1101,11110010
3	11010	10101001
4	1110,01	110010,10110100
5	100101,01	100101000,001000
6	11010,1	10101001,010100
7	1,01101	0,0111101110
8	101010	110111001,00
9	111001	1100101010,00
10	110110	1011001100,1100
11	1101010,1	101010101010,10
12	1000011	1011010010,01
13	1111,1	111100,000100
14	1100,11	100111,101001
15	110010	1001101101,00
16	10110,11	1000111,010000
17	1010,11	11000,011000
18	1111,10011	1000,1110100000
19	100010,1	100011011,10

Вариант	$(A1 + A2)_{(2)}$	$(A1 \times A2)_{(2)}$
20	100010,11	11010001,010000
21	100011,111	10100011,111100
22	1011010,1	11010110010,10
23	10000,11	110000,110000
24	1010,0111	11001,00010100
25	1001	10010,10110000
26	10100,1101	1100101,10010000
27	1000,00111	110,0011100100
28	100000,1	11110001,1000
29	1100011,11	1110111100,110000
30	100000011	10001110011010,00
31	10100000	10000101111,00
32	1101111	100101001010,00
33	10011111	1001100110010,00
34	1010111	10000111000,00
35	11010011	110011100010,00
36	11100111	1011110001110,00
37	100,1111	011,1100000000
38	10000	1000000,00
39	110111,11	11011010,111000
40	11111,11	11100001,110000

Таблица Б.5 – Ответы к практической работе № 2. Результаты операций вычитания и деления чисел в двоичной системе счисления

Вариант	$(A3 - A4)_{(2)}$	$(A3 / A4)_{(2)}$
1	1101000111	1010100,111
2	1111111111	10000000,111
3	1000111010	1011
4	101000011	1000001,10011
5	1011111000	1100000
6	1101011000	1111011,01
7	10110110	10000,0010101
8	10111111	10100,00011
9	1000110010	10001101,1
10	100001101	1111,00101
11	1111010111	1000010,10001
12	101110100	100010,1101
13	11100000	110,01010101
14	110101000	10101,0011
15	1110010000	10000011,01

Вариант	$(A3 - A4)_{(2)}$	$(A3 / A4)_{(2)}$
16	1110010111	10111000,11
17	111010001	10000,1
18	100100000	1001001
19	10001000	10010
20	111000101	1011011,10011
21	110010100	1000100,0101
22	1111101111	1000100,001
23	10111110	1000000,0101
24	11000011	10001,01
25	1001010000	1000010,11
26	1010000	101001
27	1100110010	100010001,101
28	100000000	100001
29	100001110	11001,10001
30	1010000000	10000,001111
31	100011101	101001,1011
32	1011010	100,0001101
33	1111100	10000,1
34	111001001	1011100,011
35	100101011	1000,0001111
36	111101111	100010
37	1100000111	101110,100101
38	101100010	1000111,11001
39	11010011	100100,00101
40	10000110	1000,00001101

Таблица Б.6 – Ответы к практической работе № 2. Результаты операций сложения и умножения чисел в восьмеричной системе счисления

Вариант	$(A5 + A6)_{(8)}$	$(A5 \times A6)_{(8)}$
1	354,5	3232,56
2	154,65	1050,7752
3	1,055	0,170170
4	1,304	0,351133
5	5,111	3,672744
6	1621,3	207256,44
7	4060,5	306316,30
8	47,04	370,1073
9	1,704	0,704253
10	432	15765,6727

Вариант	$(A5 + A6)_{(8)}$	$(A5 \times A6)_{(8)}$
11	410,3	4435,0007
12	772,2	3270,4727
13	10,314	7,052643
14	14,555	47,522260
15	7310,1	145202,52
16	676,26	4254,2650
17	462,51	24342,4310
18	124,02	572,7240
19	274,74	20473,7340
20	1353,55	6371,2502
21	1703,22	31622,1760
22	521,16	20646,1220
23	716,1	10224,42
24	675,67	6424,2076
25	173,42	5077,5240
26	1614,34	25342,2300
27	1104,7	33624,22
28	1506,63	25670,1734
29	476,46	13433,1751
30	1052,6	17067,0060
31	116,01	1515,1500
32	171,7	4156,04
33	620,32	5221,4160
34	165,31	1453,2730
35	27,554	176,257400
36	421,4	32250,40
37	134,002	1123,252000
38	37,54	332,1103
39	100,37	131,5254
40	506,23	16127,1000

Таблица Б.7 – Ответы к практической работе № 2. Результаты операций сложения и умножения чисел в шестнадцатеричной системе счисления

Вариант	$(A7 + A8)_{(16)}$	$(A7 \times A8)_{(16)}$
1	354,5	2C0AA68C
2	154,65	DD281F8
3	1,055	9AC4F7C
4	1,304	4D0D2E
5	5,111	16A45955

Вариант	$(A7 + A8)_{(16)}$	$(A7 \times A8)_{(16)}$
6	1621,3	27D257F
7	4060,5	4DD6BD0
8	47,04	2A4E071A
9	1,704	3ED351
10	432	BAF212D
11	410,3	B7FB70C3
12	772,2	CF1097DE
13	10,314	248867B
14	14,555	10AEA838
15	7310,1	E0BA16
16	676,26	3B2EC8
17	462,51	1EDDBA7328
18	124,02	25AC9985
19	274,74	27CE80C
20	1353,55	1DC7A6A8
21	1703,22	10100F0F0F1
22	521,16	C07401113
23	716,1	85B8B4C
24	675,67	21FFF3EA
25	173,42	CB234CC9
26	1614,34	13313252
27	1104,7	18F2DA80
28	1506,63	1500B928
29	476,46	253EC54BA
30	1052,6	DDCCDDCC
31	116,01	BEE12F4
32	171,7	AE01F5C
33	620,32	754A0B8
34	165,31	73015A8
35	27,554	9B40527ED
36	421,4	16FF80F2
37	134,002	AD80D65
38	37,54	213EAE85
39	100,37	52CCE5852
40	506,23	1A2E81678

Таблица Б.8 – Ответы к практической работе № 3. Результаты логических и арифметических операций над целыми числами

Вариант 1			
$X1_{(10)} = 45$	$X2_{(10)} = 103$	$X1\ OR\ X2_{(2)} = 01101111$	$X1\ OR\ X2_{(10)} = 111$
$X3_{(10)} = 74$		$X3\ SRL\ (5)_{(2)} = 00000010$	$X3\ SRL\ (5)_{(10)} = 2$
$X4_{(10)} = –85$	$X5_{(10)} = –13$	$[X4]_{ПК} = 11010101$	$[X5]_{ПК} = 10001101$
		$[X4]_{ОК} = 10101010$	$[X5]_{ОК} = 11110010$
		$[X4 + X5]_{ОК} = 10011101$	
		$[X4]_{ДК} = 10101011$	$[X4]_{ДК} = 10101011$
		$[X4 + X5]_{ДК} = 10011110$	
		$[X4 + X5]_{ПК} = 11100010$	
Вариант 2			
$X1_{(10)} = 30$	$X2_{(10)} = 120$	$X1\ AND\ X2_{(2)} = 00011000$	$X1\ OR\ X2_{(10)} = 24$
$X3_{(10)} = 119$		$X3\ ROL\ (6)_{(2)} = 11011101$	$X3\ ROL\ (6)_{(10)} = 221$
$X4_{(10)} = –46$	$X5_{(10)} = –6$	$[X4]_{ПК} = 10101110$	$[X5]_{ПК} = 10000110$
		$[X4]_{ОК} = 11010001$	$[X5]_{ОК} = 11111001$
		$[X4 + X5]_{ОК} = 11001011$	
		$[X4]_{ДК} = 11010010$	$[X5]_{ДК} = 11111010$
		$[X4 + X5]_{ДК} = 11001100$	
		$[X4 + X5]_{ПК} = 10110100$	
Вариант 3			
$X1_{(10)} = 39$	$X2_{(10)} = 9$	$X1\ XOR\ X2_{(2)} = 00101110$	$X1\ OR\ X2_{(10)} = 46$
$X3_{(10)} = 97$		$X3\ SRL\ (4)_{(2)} = 00000110$	$X3\ SRL\ (4)_{(10)} = 6$
$X4_{(10)} = –51$	$X5_{(10)} = 4$	$[X4]_{ПК} = 10110011$	$[X5]_{ПК} = 100$
		$[X4]_{ОК} = 11001100$	$[X5]_{ОК} = 00000100$
		$[X4 + X5]_{ОК} = 11010000$	
		$[X4]_{ДК} = 11001101$	$[X5]_{ДК} = 00000100$
		$[X4 + X5]_{ДК} = 11010001$	
		$[X4 + X5]_{ПК} = 10101111$	
Вариант 4			
$X1_{(10)} = 113$	$X2_{(10)} = 36$	$X1\ OR\ X2_{(2)} = 01110101$	$X1\ OR\ X2_{(10)} = 117$
$X3_{(10)} = 125$		$X3\ SLL\ (2)_{(2)} = 111110100$	$X3\ SLL\ (2)_{(10)} = 500$
$X4_{(10)} = –4$	$X5_{(10)} = 100$	$[X4]_{ПК} = 10000100$	$[X5]_{ПК} = 1100100$
		$[X4]_{ОК} = 11111011$	$[X5]_{ОК} = 01100100$
		$[X4 + X5]_{ОК} = 01100000$	
		$[X4]_{ДК} = 11111100$	$[X5]_{ДК} = 01100100$
		$[X4 + X5]_{ДК} = 01100000$	
		$[X4 + X5]_{ПК} = 1100000$	

<table>
<tr><td colspan="4">Вариант 5</td></tr>
<tr><td>$X1_{(10)} = 121$</td><td>$X2_{(10)} = 36$</td><td>$X1\ AND\ X2_{(2)} = 00100000$</td><td>$X1\ OR\ X2_{(10)} = 32$</td></tr>
<tr><td colspan="2">$X3_{(10)} = 60$</td><td>$X3\ ROR\ (4)_{(2)} = 11000011$</td><td>$X3\ ROR\ (4)_{(10)} = 195$</td></tr>
<tr><td rowspan="6">$X4_{(10)} = –48$</td><td rowspan="6">$X5_{(10)} = 81$</td><td>$[X4]_{ПК} = 10110000$</td><td>$[X5]_{ПК} = 1010001$</td></tr>
<tr><td>$[X4]_{ОК} = 11001111$</td><td>$[X5]_{ОК} = 01010001$</td></tr>
<tr><td colspan="2">$[X4 + X5]_{ОК} = 00100001$</td></tr>
<tr><td>$[X4]_{ДК} = 11010000$</td><td>$[X5]_{ДК} = 01010001$</td></tr>
<tr><td colspan="2">$[X4 + X5]_{ДК} = 00100001$</td></tr>
<tr><td colspan="2">$[X4 + X5]_{ПК} = 100001$</td></tr>
<tr><td colspan="4">Вариант 6</td></tr>
<tr><td>$X1_{(10)} = 123$</td><td>$X2_{(10)} = 96$</td><td>$X1\ OR\ X2_{(2)} = 01111011$</td><td>$X1\ OR\ X2_{(10)} = 123$</td></tr>
<tr><td colspan="2">$X3_{(10)} = 115$</td><td>$X3\ ROL\ (7)_{(2)} = 10111001$</td><td>$X3\ ROL\ (7)_{(10)} = 185$</td></tr>
<tr><td rowspan="6">$X4_{(10)} = 46$</td><td rowspan="6">$X5_{(10)} = 122$</td><td>$[X4]_{ПК} = 101110$</td><td>$[X5]_{ПК} = 1111010$</td></tr>
<tr><td>$[X4]_{ОК} = 00101110$</td><td>$[X5]_{ОК} = 01111010$</td></tr>
<tr><td colspan="2">$[X4 + X5]_{ОК} = 10101000$</td></tr>
<tr><td>$[X4]_{ДК} = 00101110$</td><td>$[X5]_{ДК} = 01111010$</td></tr>
<tr><td colspan="2">$[X4 + X5]_{ДК} = 10101000$</td></tr>
<tr><td colspan="2">$[X4 + X5]_{ПК} = 10101000$</td></tr>
<tr><td colspan="4">Вариант 7</td></tr>
<tr><td>$X1_{(10)} = 29$</td><td>$X2_{(10)} = 7$</td><td>$X1\ XOR\ X2_{(2)} = 00011010$</td><td>$X1\ OR\ X2_{(10)} = 26$</td></tr>
<tr><td colspan="2">$X3_{(10)} = 69$</td><td>$X3\ SLL\ (7)_{(2)} = 1000101000000$</td><td>$X3\ SLL\ (7)_{(10)} = 8832$</td></tr>
<tr><td rowspan="6">$X4_{(10)} = 6$</td><td rowspan="6">$X5_{(10)} = –30$</td><td>$[X4]_{ПК} = 110$</td><td>$[X5]_{ПК} = 10011110$</td></tr>
<tr><td>$[X4]_{ОК} = 00000110$</td><td>$[X5]_{ОК} = 11100001$</td></tr>
<tr><td colspan="2">$[X4 + X5]_{ОК} = 11100111$</td></tr>
<tr><td>$[X4]_{ДК} = 00000110$</td><td>$[X5]_{ДК} = 11100010$</td></tr>
<tr><td colspan="2">$[X4 + X5]_{ДК} = 11101000$</td></tr>
<tr><td colspan="2">$[X4 + X5]_{ПК} = 10011000$</td></tr>
<tr><td colspan="4">Вариант 8</td></tr>
<tr><td>$X1_{(10)} = 15$</td><td>$X2_{(10)} = 65$</td><td>$X1\ OR\ X2_{(2)} = 01001111$</td><td>$X1\ OR\ X2_{(10)} = 79$</td></tr>
<tr><td colspan="2">$X3_{(10)} = 49$</td><td>$X3\ SRL\ (6)_{(2)} = 00000000$</td><td>$X3\ SRL\ (6)_{(10)} = 0$</td></tr>
<tr><td rowspan="6">$X4_{(10)} = –77$</td><td rowspan="6">$X5_{(10)} = 105$</td><td>$[X4]_{ПК} = 11001101$</td><td>$[X5]_{ПК} = 1101001$</td></tr>
<tr><td>$[X4]_{ОК} = 10110010$</td><td>$[X5]_{ОК} = 01101001$</td></tr>
<tr><td colspan="2">$[X4 + X5]_{ОК} = 00011100$</td></tr>
<tr><td>$[X4]_{ДК} = 10110011$</td><td>$[X5]_{ДК} = 01101001$</td></tr>
<tr><td colspan="2">$[X4 + X5]_{ДК} = 00011100$</td></tr>
<tr><td colspan="2">$[X4 + X5]_{ПК} = 11100$</td></tr>
</table>

<table>
<tr><td colspan="4">Вариант 9</td></tr>
<tr><td>$X1_{(10)} = 120$</td><td>$X2_{(10)} = 49$</td><td>$X1\ AND\ X2_{(2)} = 00110000$</td><td>$X1\ OR\ X2_{(10)} = 48$</td></tr>
<tr><td colspan="2">$X3_{(10)} = 9$</td><td>$X3\ SLL\ (5)_{(2)} = 100100000$</td><td>$X3\ SLL\ (5)_{(10)} = 288$</td></tr>
<tr><td rowspan="6">$X4_{(10)} = –76$</td><td rowspan="6">$X5_{(10)} = 6$</td><td>$[X4]_{ПК} = 11001100$</td><td>$[X5]_{ПК} = 110$</td></tr>
<tr><td>$[X4]_{ОК} = 10110011$</td><td>$[X5]_{ОК} = 00000110$</td></tr>
<tr><td colspan="2">$[X4 + X5]_{ОК} = 10111001$</td></tr>
<tr><td>$[X4]_{ДК} = 10110100$</td><td>$[X5]_{ДК} = 00000110$</td></tr>
<tr><td colspan="2">$[X4 + X5]_{ДК} = 10111010$</td></tr>
<tr><td colspan="2">$[X4 + X5]_{ПК} = 11000110$</td></tr>
<tr><td colspan="4">Вариант 10</td></tr>
<tr><td>$X1_{(10)} = 41$</td><td>$X2_{(10)} = 10$</td><td>$X1\ XOR\ X2_{(2)} = 00100011$</td><td>$X1\ OR\ X2_{(10)} = 35$</td></tr>
<tr><td colspan="2">$X3_{(10)} = 44$</td><td>$X3\ SLL\ (7)_{(2)} = 1011000000000$</td><td>$X3\ SLL\ (7)_{(10)} = 5632$</td></tr>
<tr><td rowspan="6">$X4_{(10)} = –68$</td><td rowspan="6">$X5_{(10)} = –51$</td><td>$[X4]_{ПК} = 11000100$</td><td>$[X5]_{ПК} = 10110011$</td></tr>
<tr><td>$[X4]_{ОК} = 10111011$</td><td>$[X5]_{ОК} = 11001100$</td></tr>
<tr><td colspan="2">$[X4 + X5]_{ОК} = 10001000$</td></tr>
<tr><td>$[X4]_{ДК} = 10111100$</td><td>$[X5]_{ДК} = 11001101$</td></tr>
<tr><td colspan="2">$[X4 + X5]_{ДК} = 10001001$</td></tr>
<tr><td colspan="2">$[X4 + X5]_{ПК} = 11110111$</td></tr>
<tr><td colspan="4">Вариант 11</td></tr>
<tr><td>$X1_{(10)} = 102$</td><td>$X2_{(10)} = 100$</td><td>$X1\ AND\ X2_{(2)} = 01100100$</td><td>$X1\ OR\ X2_{(10)} = 100$</td></tr>
<tr><td colspan="2">$X3_{(10)} = 89$</td><td>$X3\ SRL\ (5)_{(2)} = 00000010$</td><td>$X3\ SRL\ (5)_{(10)} = 2$</td></tr>
<tr><td rowspan="6">$X4_{(10)} = 57$</td><td rowspan="6">$X5_{(10)} = 16$</td><td>$[X4]_{ПК} = 111001$</td><td>$[X5]_{ПК} = 10000$</td></tr>
<tr><td>$[X4]_{ОК} = 00111001$</td><td>$[X5]_{ОК} = 00010000$</td></tr>
<tr><td colspan="2">$[X4 + X5]_{ОК} = 01001001$</td></tr>
<tr><td>$[X4]_{ДК} = 00111001$</td><td>$[X5]_{ДК} = 00010000$</td></tr>
<tr><td colspan="2">$[X4 + X5]_{ДК} = 01001001$</td></tr>
<tr><td colspan="2">$[X4 + X5]_{ПК} = 1001001$</td></tr>
<tr><td colspan="4">Вариант 12</td></tr>
<tr><td>$X1_{(10)} = 37$</td><td>$X2_{(10)} = 11$</td><td>$X1\ OR\ X2_{(2)} = 00101111$</td><td>$X1\ OR\ X2_{(10)} = 47$</td></tr>
<tr><td colspan="2">$X3_{(10)} = 118$</td><td>$X3\ ROR\ (3)_{(2)} = 11001110$</td><td>$X3\ ROR\ (3)_{(10)} = 206$</td></tr>
<tr><td rowspan="6">$X4_{(10)} = 106$</td><td rowspan="6">$X5_{(10)} = –92$</td><td>$[X4]_{ПК} = 1101010$</td><td>$[X5]_{ПК} = 11011100$</td></tr>
<tr><td>$[X4]_{ОК} = 01101010$</td><td>$[X5]_{ОК} = 10100011$</td></tr>
<tr><td colspan="2">$[X4 + X5]_{ОК} = 00001110$</td></tr>
<tr><td>$[X4]_{ДК} = 01101010$</td><td>$[X5]_{ДК} = 10100100$</td></tr>
<tr><td colspan="2">$[X4 + X5]_{ДК} = 00001110$</td></tr>
<tr><td colspan="2">$[X4 + X5]_{ПК} = 1110$</td></tr>
</table>

Вариант 13			
$X1_{(10)} = 51$	$X2_{(10)} = 10$	$X1\ XOR\ X2_{(2)} = 00111001$	$X1\ OR\ X2_{(10)} = 57$
$X3_{(10)} = 105$		$X3\ ROR\ (5)_{(2)} = 01001011$	$X3\ ROR\ (5)_{(10)} = 75$
$X4_{(10)} = 47$	$X5_{(10)} = –111$	$[X4]_{ПК} = 101111$	$[X5]_{ПК} = 11101111$
		$[X4]_{ОК} = 00101111$	$[X5]_{ОК} = 10010000$
		$[X4 + X5]_{ОК} = 10111111$	
		$[X4]_{ДК} = 00101111$	$[X5]_{ДК} = 10010001$
		$[X4 + X5]_{ДК} = 11000000$	
		$[X4 + X5]_{ПК} = 11000000$	
Вариант 14			
$X1_{(10)} = 35$	$X2_{(10)} = 39$	$X1\ OR\ X2_{(2)} = 00100111$	$X1\ OR\ X2_{(10)} = 39$
$X3_{(10)} = 33$		$X3\ SLL\ (3)_{(2)} = 100001000$	$X3\ SLL\ (3)_{(10)} = 264$
$X4_{(10)} = 69$	$X5_{(10)} = –21$	$[X4]_{ПК} = 1000101$	$[X5]_{ПК} = 10010101$
		$[X4]_{ОК} = 01000101$	$[X5]_{ОК} = 11101010$
		$[X4 + X5]_{ОК} = 00110000$	
		$[X4]_{ДК} = 01000101$	$[X5]_{ДК} = 11101011$
		$[X4 + X5]_{ДК} = 00110000$	
		$[X4 + X5]_{ПК} = 110000$	
Вариант 15			
$X1_{(10)} = 107$	$X2_{(10)} = 41$	$X1\ AND\ X2_{(2)} = 00101001$	$X1\ OR\ X2_{(10)} = 41$
$X3_{(10)} = 72$		$X3\ SRL\ (5)_{(2)} = 00000010$	$X3\ SRL\ (5)_{(10)} = 2$
$X4_{(10)} = 85$	$X5_{(10)} = 125$	$[X4]_{ПК} = 1010101$	$[X5]_{ПК} = 1111101$
		$[X4]_{ОК} = 01010101$	$[X5]_{ОК} = 01111101$
		$[X4 + X5]_{ОК} = 11010010$	
		$[X4]_{ДК} = 01010101$	$[X5]_{ДК} = 01111101$
		$[X4 + X5]_{ДК} = 11010010$	
		$[X4 + X5]_{ПК} = 11010010$	

Таблица Б.9 – Ответы к практической работе № 3. Результаты операций над модифицированным двоичным кодом числа

Вариант	***X*6**	**Код**	**Сдвиг**	**Ответ**	
				[МК]	**число**
1	10.010011	МПК	*SAR* (3)	11.110010	–50
2	10.010001	МПК	*SAR* (7)	10.000000	0
3	11.110101	МДК	*SAL* (3)	11.101000	–24
4	00.010011	МПК	*SAL* (5)	00.100000	32
5	11.111110	МОК	*SAL* (4)	11.100000	–31
6	10.010101	МДК	*SAR* (2)	11.110101	–11

Вариант	*X*6	Код	Сдвиг	Ответ	
				[МК]	число
7	10.000011	МОК	*SAR* (6)	11.111111	–0
8	10.110011	МДК	*SAR* (10)	11.111111	–1
9	00.000110	МПК	*SAL* (4)	00.100000	32
10	00.000010	МОК	*SAL* (5)	01.000000	0
11	11.010001	МДК	*SAL* (2)	11.000100	–60
12	00.000110	МОК	*SAL* (3)	00.110000	48
13	00.000111	МДК	*SAL* (4)	00.110000	48
14	10.010011	МПК	*SAR* (5)	11.111100	–28
15	10.011011	МОК	*SAR* (8)	11.111111	–0

Таблица Б.10 – Ответы к практической работе № 4. Результаты операций над вещественными числами в формате с плавающей точкой (запятой)

Вари-ант	$Y_{SINGLE(2)}$	$S_{E(2)}$	$S_{M(2)}$	$P_{E(2)}$	$P_{M(2)}$	$Q_{E(2)}$	$Q_{M(2)}$
1	01000011001010001011111101111101	1.011	1.100000	1.001	1.100000	1.001	0.1111
2	11000011101000110000111010111000	0.010	0.111010	0.000	0.101000	1.011	1.1100
3	11000011011010001100000011000101	0.000	0.110000	0.001	1.101000	0.011	1.1010
4	01000011010001000111010001111011	1.001	0.101010	1.101	0.111000	1.001	0.1000
5	11000011000001001100100011110110	0.010	1.100110	0.000	0.101110	0.100	0.1100
6	01000011001110111010000000000000	1.001	1.100000	0.100	1.100100	1.100	1.1001
7	01000011000101011100000011000101	0.000	0.110100	1.011	1.111010	0.100	1.1000
8	01000011101010101110001111111000	0.011	0.101010	1.001	0.110000	1.011	0.0110
9	01000011011101010011111011111010	0.001	1.101000	1.001	0.110000	1.011	0.1011
10	01000011010101100101001000101101	0.001	1.111110	1.001	1.100100	0.110	1.1010
11	11000011011101001100100111111100	0.100	1.100011	0.001	1.111100	0.010	1.1001
12	11000011001011111101101000011101	0.010	1.110010	0.010	0.100001	0.100	0.1000
13	11000011001000100101110100101111	0.010	1.100010	1.001	0.101101	1.010	0.0101
14	01000011100101111001010001011010	1.001	0.100000	1.111	0.110100	0.010	1.1110
15	11000011010101001000100111111100	1.010	0.100000	1.110	1.100100	1.100	1.1100

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ И РЕКОМЕНДОВАННОЙ ЛИТЕРАТУРЫ

1 IEEE Std 754–2019. IEEE Standard for Floating-Point Arithmetic 754–2019. – Appr. 13.06.2019. – New York : IEEE Computer Society.

2 Савенко, А. Г. Основы компьютерной техники. Практические занятия : учеб.-метод. пособие / А. Г. Савенко, А. В. Матвеев. – Минск : БГУИР, 2020. – 82 с.

3 Куприянова, Д. В. Арифметические и логические основы вычислительной техники : пособие / Д. В. Куприянова, И. В. Лукьянова, Ю. А. Луцик. – Минск : БГУИР, 2021. – 72 с.

4 Луцик, Ю. А. Арифметические и логические основы вычислительной техники : учеб. пособие / Ю. А. Луцик, И. В. Лукьянова. – Минск : БГУИР, 2014. – 174 с.

5 Фомин, Д. В. Основы компьютерной электроники : учеб. пособие / Д. В. Фомин. – М.-Берлин : ДиректМедиа, 2014. – 108 с.

6 Сергеев, Н. П. Основы вычислительной техники : учеб. пособие для вузов / Н. П. Сергеев, Н. П. Вашкевич. – 2-е изд., перераб. и доп. – М. : Высш. шк., 1988. – 311 с.

7 Таненбаум, Э. Архитектура компьютера / Э. Таненбаум, Т. Остин. – 6-е изд. – СПб. : Питер, 2013. – 816 с.

8 Питолин, В. Е. Архитектура ПЭВМ и вычислительных систем : пособие / В. Е. Питолин, С. Г. Сурто. – Новополоцк : ПГУ, 2022. – 286 с.

9 Буза, М. К. Архитектура компьютеров : учеб. / М. К. Буза. – Минск : Новое знание, 2007. – 559 с.

10 Старков, В. В. Азбука персонального компьютера. Архитектура, устройство и конфигурирование / В. В. Старков. – М. : Горячая линия – Телеком, 2000. – 228 с.

11 Мейлахс, А. Л. Практикум по математическим основам информатики. Ч. 1 : Системы счисления. Двоичная арифметика. Представление чисел в памяти ЭВМ / А. Л. Мейлахс. – М. : Изд-во МГГУ, 2004. – 63 с.

12 Пешков, А. Т. Организация и функционирование ЭВМ : метод. пособие для студентов специальности «Программное обеспечение информационных технологий» дневной формы обучения. В 3 ч. Ч. 1 : Арифметические основы ЭВМ / А. Т. Пешков. – Минск : БГУИР, 2004. – 55 с.

13 Андреева, Е. Н. Системы счисления и компьютерная арифметика / Е. Н. Андреева, И. Н. Фалина. – 2-е изд. – М. : Лаборатория Базовых Знаний, 2000. – 248 с.

14 Гашков, С. Б. Системы счисления и их применение / С. Б. Гашков. – Минск : МЦНМО, 2004. – 52 с.

15 Поснов, Н. Н. Арифметика вычислительных машин в упражнениях и задачах : системы счисления, коды / Н. Н. Поснов. – Минск : Университетское, 1984. – 221 с.

16 Петцольд, Ч. Код / Ч. Петцольд ; под общ. ред. Д. З. Вибе. – М. : Рус. ред., 2001. – 521 с.

Св. план 2024, поз. 37

Учебное издание

Савенко Андрей Геннадьевич
Парамонов Антон Иванович

ОСНОВЫ КОМПЬЮТЕРНОЙ ТЕХНИКИ

УЧЕБНО-МЕТОДИЧЕСКОЕ ПОСОБИЕ

Редактор *А. Ю. Шурко*
Корректор *Е. Н. Батурчик*
Компьютерная правка, оригинал-макет *А. А. Лущикова*

Подписано в печать 18.06.2025. Формат 60×84 1/16. Бумага офсетная. Гарнитура «Таймс».
Отпечатано на ризографе. Усл. печ. л. 7,79. Уч.-изд. л. 8,0. Тираж 60 экз. Заказ 143.

Издатель и полиграфическое исполнение: учреждение образования
«Белорусский государственный университет информатики и радиоэлектроники».
Свидетельство о государственной регистрации издателя, изготовителя,
распространителя печатных изданий №1/238 от 24.03.2014,
№2/113 от 07.04.2014, №3/615 от 07.04.2014.
Ул. П. Бровки, 6, 220013, г. Минск