МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное
учреждение высшего образования
«СЕВАСТОПОЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Кафедра информационных технологий и компьютерных систем

АРИФМЕТИЧЕСКИЕ ОСНОВЫ КОМПЬЮТЕРОВ. ПРЕДСТАВЛЕНИЕ И ОБРАБОТКА ДАННЫХ ПРОСТЫХ ТИПОВ В ЭВМ

Методические указания
к выполнению цикла практических работ
по дисциплине «Алгоритмизация
и программирование вычислительных процессов»
для студентов направления
44.03.05 - Педагогическое образование
(с двумя профилями подготовки. Профиль: математика и информатика)
дневной формы обучения

Севастополь
СевГУ
2023

УДК 004.421.2:511.11(076.5)
ББК 32.973-02я7
А817

Р е ц е н з е н т :
В. Ю. Карлусов - канд. техн. наук, доцент кафедры информационных систем

Составители: Т. В. Волкова, Е. С. Владимирова

А817 Арифметические основы компьютеров. Представление и обработка данных простых типов в ЭВМ : методические указания к выполнению цикла практических работ по дисциплине «Алгоритмизация и программирование вычислительных процессов» для студентов направления 44.03.05 – Педагогическое образование (с двумя профилями подготовки. Профиль: математика и информатика) дневной формы обучения / Севастопольский государственный университет, Институт информационных технологий и управления в технических системах, кафедра информационных технологий и компьютерных систем ; сост. Т. В. Волкова, Е. С. Владимирова. – Севастополь : СевГУ, 2023. – 58 с. – Текст : электронный.

Целью методических указаний является оказание помощи студентам при выполнении самостоятельных практических работ по дисциплине «Алгоритмизация и программирование вычислительных процессов». Методические указания предназначены для студентов первого курса дневной формы обучения направления 44.03.05 – Педагогическое образование (с двумя профилями подготовки. Профиль: математика и информатика). Последовательное выполнение пяти практических работ позволит студентам закрепить знания по темам «Системы счисления», «Системы счисления, используемые в программировании», «Типы данных для представления целых чисел в ЭВМ. Выполнение операций над целыми числами», «Представление вещественных чисел в ЭВМ. Типы данных с плавающей точкой. Выполнение операций над вещественными числами», «Обработка двоичных векторов. Поразрядные логические операции. Операции сдвигов. Операции сравнения и булевой логики». Для каждого из 18 заданий, предложенных в методических указаниях, разработано 30 вариантов, даны ссылки на литературные источники, с которыми студент должен ознакомиться при подготовке к выполнению заданий, для каждого задания приводится пример (примеры) выполнения с краткими пояснениями. Для каждой практической работы разработаны контрольные вопросы для проверки усвоения студентом соответствующего материала.

УДК 004.421.2:511.11(076.5)
ББК 32.973-02я73

Рассмотрены и рекомендованы к изданию на заседании кафедры информационных технологий и компьютерных систем, протокол № 2 от 14 сентября 2022 г.

© Волкова Т. В., Владимирова Е. С., сост., 2023
© ФГАОУ ВО «Севастопольский государственный университет», 2023

СОДЕРЖАНИЕ

1. ОБЩИЕ ТРЕБОВАНИЯ К ВЫПОЛНЕНИЮ ЦИКЛА ПРАКТИЧЕСКИХ РАБОТ 4

2. ТЕОРЕТИЧЕСКАЯ ПОДГОТОВКА 4

3. ВЫБОР ВАРИАНТА ЗАДАНИЯ 5

4. ПРАКТИЧЕСКАЯ РАБОТА 1. СИСТЕМЫ СЧИСЛЕНИЯ. ЗАДАНИЯ 1 – 3 6

5. ПРАКТИЧЕСКАЯ РАБОТА 2. СИСТЕМЫ СЧИСЛЕНИЯ, ИСПОЛЬЗУЕМЫЕ В ПРОГРАММИРОВАНИИ. ЗАДАНИЯ 4 – 6 11

6. ПРАКТИЧЕСКАЯ РАБОТА 3. ТИПЫ ДАННЫХ ДЛЯ ПРЕДСТАВЛЕНИЯ ЦЕЛЫХ ЧИСЕЛ В ЭВМ. ВЫПОЛНЕНИЕ ОПЕРАЦИЙ НАД ЦЕЛЫМИ ЧИСЛАМИ. ЗАДАНИЯ 7 – 10 14

7. ПРАКТИЧЕСКАЯ РАБОТА 4. ПРЕДСТАВЛЕНИЕ ВЕЩЕСТВЕННЫХ ЧИСЕЛ В ЭВМ. ТИПЫ ДАННЫХ С ПЛАВАЮЩЕЙ ТОЧКОЙ. ВЫПОЛНЕНИЕ ОПЕРАЦИЙ НАД ВЕЩЕСТВЕННЫМИ ЧИСЛАМИ. ЗАДАНИЯ 11 – 13 23

8. ПРАКТИЧЕСКАЯ РАБОТА 5. ОБРАБОТКА ДВОИЧНЫХ ВЕКТОРОВ. ПОРАЗРЯДНЫЕ ЛОГИЧЕСКИЕ ОПЕРАЦИИ. ОПЕРАЦИИ СДВИГОВ. ОПЕРАЦИИ СРАВНЕНИЯ И БУЛЕВОЙ ЛОГИКИ. ЗАДАНИЯ 14 – 18 42

БИБЛИОГРАФИЧЕСКИЙ СПИСОК 57

ПРИЛОЖЕНИЕ А 58

1. ОБЩИЕ ТРЕБОВАНИЯ К ВЫПОЛНЕНИЮ ПРАКТИЧЕСКИХ РАБОТ

1.1. Цель работы

Целью выполнения цикла практических работ является закрепление знаний по темам «Системы счисления», «Системы счисления, используемые в программировании», «Типы данных для представления целых чисел в ЭВМ. Выполнение операций над целыми числами», «Представление вещественных чисел в ЭВМ. Типы данных с плавающей точкой. Выполнение операций над вещественными числами», «Обработка двоичных векторов. Поразрядные логические операции. Операции сдвигов. Операции сравнения и булевой логики».

1.2. Требования к отчету

Отчет о выполнении цикла практических работ представляется в тетради в клетку или на листах формата А4. Первый лист отчета – титульный. Пример оформления титульного листа приведен в приложении А. По каждой практической работе отчет должен содержать следующие разделы:

1) название работы
2) цель работы;
3) выполнение задания (в данном разделе должны быть приведены условия заданий согласно варианту и подробные решения этих заданий с пояснениями);
4) вывод.

2. ТЕОРЕТИЧЕСКАЯ ПОДГОТОВКА

Задания с 1 по 3 составляют первую практическую работу, предназначенную для закрепления знаний студентов по теме: «Системы счисления».

Задания с 4 по 6 составляют вторую практическую работу, предназначенную для закрепления знаний студентов по теме: «Системы счисления, используемые в программировании».

Задания с 7 по 10 составляют третью практическую работу, предназначенную для закрепления знаний студентов по теме: «Типы данных для представления целых чисел в ЭВМ. Выполнение операций над целыми числами».

Задания с 11 по 13 составляют четвертую практическую работу, предназначенную для закрепления знаний студентов по теме: «Представление вещественных чисел в ЭВМ. Типы данных с плавающей точкой. Выполнение операций над вещественными числами».

Задания с 14 по 18 составляют пятую практическую работу, предназначенную для закрепления знаний студентов по темам: «Обработка двоичных векторов. Поразрядные логические операции. Операции сдвигов.

Операции сравнения и булевой логики», «Логические выражения», «Ошибки в программах, вызванные некорректным использованием типов данных».

Перед выполнением первой, второй и третьей практических работ студент должен изучить соответствующие темы [1] (с. 22-25), [2] (с. 6-24), [3] (с. 36-56, с. 61-62, с. 64-65), [4] (с. 53-56, с. 82-88).

С теоретическим материалом, используемым в четвертой практической работе, можно ознакомиться в [2] (с. 25-32), [3] (с. 62-64, с. 74-77, с.88-97), [4] (с. 56-58, с. 88-97), [5].

Перед выполнением пятой практической работы рекомендуется ознакомиться с [4], с.58-63, с. 98-105, [6].

3. ВЫБОР ВАРИАНТА ЗАДАНИЯ

Номер варианта (НВ) вычисляется по следующему правилу:

НВ = АВ % 30,

где АВ – две последние цифры номера зачетной книжки студента,

% – операция получения остатка от целочисленного деления.

Если АВ = 00, студент выполняет вариант № 30.

4. ПРАКТИЧЕСКАЯ РАБОТА 1. СИСТЕМЫ СЧИСЛЕНИЯ. ЗАДАНИЯ 1 – 3

4.1. Содержание, варианты и примеры выполнения заданий

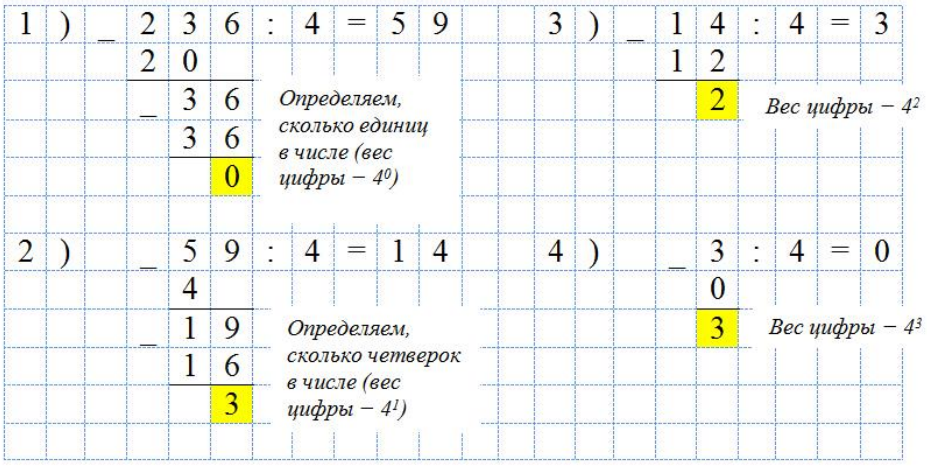
Задание 1. Перевести целое число X из системы счисления с основанием P=10 в систему счисления с основанием Q. Сделать проверку путем обратного перевода в десятичную систему методом построения степенного ряда. Варианты задания приведены в таблице 1.

Таблица 1 – Варианты для задания 1

НВ	X (P=10)	Q	НВ	X (P=10)	Q
1	8427	3	16	5929	3
2	8427	4	17	5929	4
3	8427	5	18	5929	5
4	8427	6	19	5929	6
5	8427	7	20	5929	7
6	5156	3	21	7456	3
7	5156	4	22	7456	4
8	5156	5	23	7456	5
9	5156	6	24	7456	6
10	5156	7	25	7456	7
11	4981	3	26	6798	3
12	4981	4	27	6798	4
13	4981	5	28	6798	5
14	4981	6	29	6798	6
15	4981	7	30	6798	7

Правило перевода целых чисел из системы счисления с основанием P в систему счисления с основанием Q. *Перевод целого числа из системы счисления с основанием P в систему счисления с основанием Q осуществляется путем деления числа на Q, представленное в системе счисления с основанием P. В ходе деления вычисляется частное и остаток от деления, который и является очередной цифрой результата. Затем частное снова делится на Q. Процесс деления продолжается до тех пор, пока частное не станет равным нулю.*

Пример выполнения задания 1. Перевести целое число X из системы счисления с основанием P=10 в систему счисления с основанием Q. Сделать проверку путем обратного перевода. $X=236_{10}$, Q=4.



$X=236_{10}=3230_4$.

Проверка: $0\cdot 4^0+3\cdot 4^1+2\cdot 4^2+3\cdot 4^3=12+32+3\cdot 64=236.$

Задание 2. Перевести правильную дробь X из системы счисления с основанием P=10 в систему счисления с основанием Q с точностью $\varepsilon = Q^{-5}$. Сделать проверку путем обратного перевода в десятичную систему методом построения степенного ряда. Варианты задания приведены в таблице 2.

Таблица 2 – Варианты для задания 2

НВ	Число X (P=10)	Q	НВ	Число X (P=10)	Q
1	0,842	3	16	0,592	3
2	0,842	4	17	0,592	4
3	0,842	5	18	0,592	5
4	0,842	6	19	0,592	6
5	0,842	7	20	0,592	7
6	0,515	3	21	0,745	3
7	0,515	4	22	0,745	4
8	0,515	5	23	0,745	5
9	0,515	6	24	0,745	6
10	0,515	7	25	0,745	7
11	0,498	3	26	0,679	3
12	0,498	4	27	0,679	4
13	0,498	5	28	0,679	5
14	0,498	6	29	0,679	6
15	0,498	7	30	0,679	7

При переводе вещественных чисел из системы счисления с основанием P в систему счисления с основанием Q выполняется аппроксимация результата с заданной точностью. Точность **ε** обычно задается в виде отрицательной степени основания системы счисления, в которую осуществляется перевод (например, перевести действительное число из двоичной системы в десятичную с точностью до $0{,}001=10^{-3}$).

Правильная дробь − вещественное число с нулевой целой частью.

Правило перевода правильных дробей из системы счисления с основанием P в систему счисления с основанием Q.

1) *Правильную дробь, записанную в системе счисления P, умножают на Q, записанное в системе счисления P.*
2) *Целую часть результата записывают как старшую цифру после запятой в системе счисления Q.*
3) *Дробную часть результата вновь умножают на Q.*
4) *Пункты 2 и 3 повторяют либо до получения нулевой дробной части, либо до достижения требуемой точности.*

Пример выполнения задания 2. Перевести число $X_{10}=0{,}236$ в четверичную систему счисления с точностью $\varepsilon = 4^{-5}$.

0,		2	3	6
	x			4
0		9	4	4
	x			4
3		7	7	6
	x			4
3		1	0	4
	x			4
0		4	1	6
	x			4
1		6	6	4

Проверка:

$$0{,}03301_4 = 0\times4^{-1}+3\times4^{-2}+3\times4^{-3}+0\times4^{-4}+1\times4^{-5}=$$

$$=\frac{3\times4^3+3\times4^2+1}{4^5}=\frac{241}{1024}=0{,}235_{10}$$

Результат подтверждает, что перевод приближенный.

Задание 3. Перевести целое положительное число X из двоичной системы счисления в восьмеричную, шестнадцатеричную и десятичную системы счисления. Варианты задания приведены в таблице 3.

Таблица 3 – Варианты для задания 3

НВ	Двоичное число X	НВ	Двоичное число X	НВ	Двоичное число X
1	10101111101100101	11	1110000110101000	21	10101100000010100
2	10110111110011110	12	1001111101010010	22	1100011101000110
3	1100001110101101	13	1000101100111101	23	1101001011010110
4	1100001110101110	14	1110111101010111	24	1111000010100001
5	10111100011011110	15	1100111101111000	25	1110000100001001
6	1110010111010111	16	10101001100011110	26	1000111000101010
7	10101011010111101	17	1110011010010110	27	10101010101010
8	10010011111000100	18	1101011100110001	28	1100110111101100
9	10000011111101111	19	1100111000010010	29	1110111011101110
10	11110111110100110	20	1110001100010011	30	1100111011111001

Наиболее просто осуществляется перевод чисел из восьмеричной и шестнадцатеричной систем счисления в двоичную и обратно, так как и 8 и 16 являются степенями двойки.

$8 = 2^3$, поэтому восьмеричная цифра представляется тремя двоичными разрядами (с весами соответственно 4,2,1) – таблица 4.

$16 = 2^4$, поэтому шестнадцатеричная цифра представляется четырьмя двоичными разрядами (с весами соответственно 8,4,2,1) – таблица 5.

В десятичную систему из двоичной, восьмеричной или шестнадцатеричной системы число можно перевести методом построения степенного ряда.

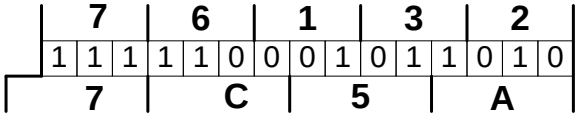
Таблица 4 – Двоичные эквиваленты восьмеричных цифр

	2^2	2^1	2^0
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

Таблица 5 – Двоичные эквиваленты шестнадцатеричных цифр

	2^3	2^2	2^1	2^0
	8	**4**	**2**	**1**
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
A	1	0	1	0
B	1	0	1	1
C	1	1	0	0
D	1	1	0	1
E	1	1	1	0
F	1	1	1	1

Пример выполнения задания 3. Перевести целое положительное число X из двоичной системы счисления в восьмеричную, шестнадцатеричную и десятичную системы счисления. $X=111110001011010_2$



$$76132_8=2*8^0+3*8^1+1*8^2+6*8^3+7*8^4=31834$$
$$7C5A_{16}=10*16^0+5*16^1+12*16^2+7*16^3=31834$$

4.2. Контрольные вопросы

1) Какие системы счисления называют позиционными весомозначными? Приведите примеры.
2) Какие цифры существуют в четверичной, шестеричной, восьмеричной, шестнадцатеричной системах счисления? Сколько весит старший разряд 5-значного пятеричного числа?
2) По какому правилу осуществляется перевод целого числа из системы счисления с основанием P в систему счисления с основанием Q?
3) По какому правилу осуществляется перевод правильной дроби из системы счисления с основанием P в систему счисления с основанием Q?
4) Как перевести число из системы счисления с основанием P в десятичную систему счисления? Проиллюстрируйте применение метода составления степенного ряда на примере перевода чисел $23FD,B9_{16}$; $753,672_8$; $101001,101011_2$ в десятичную систему счисления.
5) Сколько двоичных разрядов (битов) соответствует восьмеричной цифре и почему? Назовите двоичные эквиваленты восьмеричных цифр.
6) Сколько двоичных разрядов соответствует шестнадцатеричной цифре и почему? Назовите двоичные эквиваленты шестнадцатеричных цифр.
7) Как осуществляется перевод двоичного числа в восьмеричную систему счисления и обратно?
8) Как осуществляется перевод двоичного числа в шестнадцатеричную систему счисления и обратно?

5. ПРАКТИЧЕСКАЯ РАБОТА 2. СИСТЕМЫ СЧИСЛЕНИЯ, ИСПОЛЬЗУЕМЫЕ В ПРОГРАММИРОВАНИИ. ЗАДАНИЯ 4 – 6

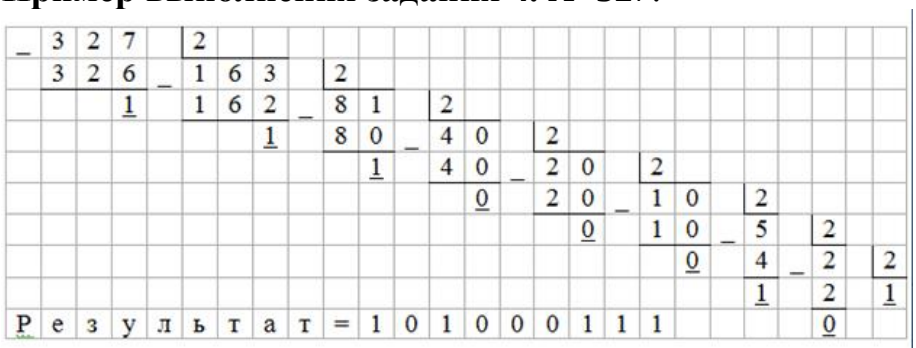
5.1. Содержание, варианты и примеры выполнения заданий

Задание 4. Перевести целое положительное число X из десятичной системы счисления в двоичную. Сделать проверку путем обратного перевода в десятичную систему методом построения степенного ряда. Варианты задания приведены в таблице 6.

Таблица 6 – Варианты для задания 4

НВ	X	НВ	X
1	425	16	595
2	450	17	612
3	473	18	624
4	495	19	638
5	462	20	645
6	437	21	657
7	416	22	664
8	521	23	675
9	530	24	680
10	545	25	692
11	559	26	715
12	564	27	722
13	572	28	748
14	579	29	759
15	584	30	761

Пример выполнения задания 4. X=327.



Проверка:

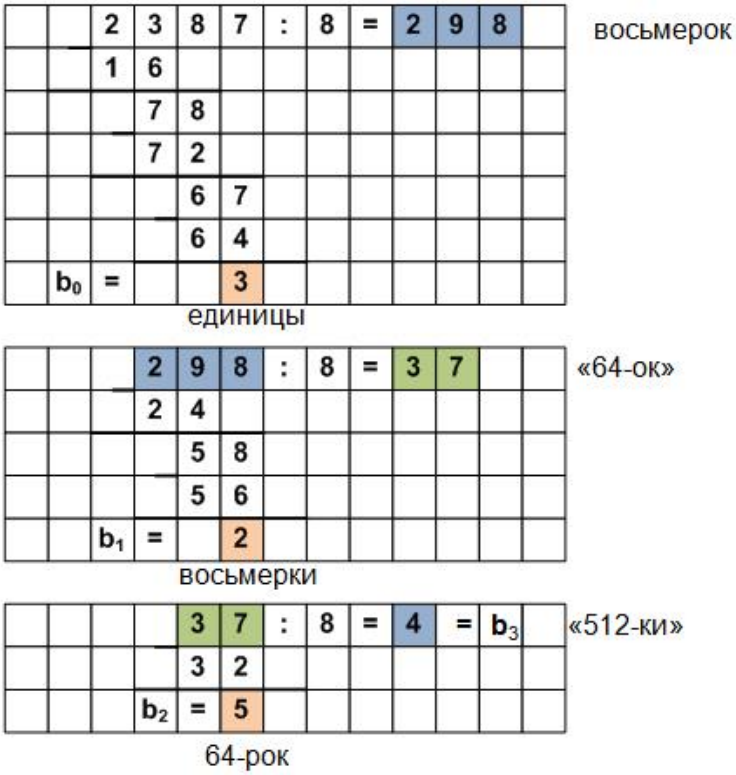
$$2^0 + 2^1 + 2^2 + 2^6 + 2^8 = 1 + 2 + 4 + 64 + 256 = 327$$

Задание 5. Перевести целое положительное число X из десятичной системы счисления в двоичную через восьмеричную. Сделать проверку путем обратного перевода из восьмеричной в десятичную систему счисления методом построения степенного ряда. Варианты задания представлены в таблице 7.

Таблица 7 – Варианты для заданий 5 и 6

НВ	Число X	НВ	Число X
1	5125	16	6652
2	5250	17	6725
3	5341	18	6812
4	5422	19	6935
5	5594	20	7055
6	5631	21	7123
7	5725	22	7257
8	5874	23	7348
9	5918	24	7421
10	6052	25	8539
11	6174	26	8640
12	6279	27	8745
13	6358	28	8801
14	6435	29	8935
15	6521	30	9034

Пример выполнения задания 5. X=2387.



Результат равен 4523_8.

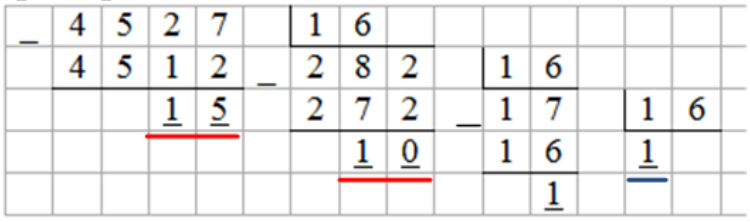
Проверка:

$3 \cdot 8^0 + 2 \cdot 8^1 + 5 \cdot 8^2 + 4 \cdot 8^3 = 3 \cdot 1 + 2 \cdot 8 + 5 \cdot 64 + 4 \cdot 512 = 3 + 16 + 320 + 2048 = 2387_{10}$

Перевод в двоичную систему: $4523_8 = 100\ 101\ 010\ 011_2$

Задание 6. Перевести целое положительное число X из десятичной системы в двоичную через шестнадцатеричную. Сделать проверку путем обратного перевода из шестнадцатеричной в десятичную систему счисления методом построения степенного ряда. Варианты задания представлены в таблице 7.

Пример выполнения задания 6. X=327.



$4527_{10}=11AF_{16}=0001\ 0001\ 1010\ 1111_2 = 1\ 0001\ 1010\ 1111_2$

Проверка:

$11AF_{16}=15 \cdot 16^0+10 \cdot 16^1+1 \cdot 16^2+1 \cdot 16^3=15+160+256+4096=4527_{10}$

5.2. Контрольные вопросы

1) Сколько двоичных разрядов (битов) соответствует восьмеричной цифре и почему? Назовите двоичные эквиваленты восьмеричных цифр.
2) Сколько двоичных разрядов соответствует шестнадцатеричной цифре и почему? Назовите двоичные эквиваленты шестнадцатеричных цифр.
3) Как осуществляется перевод двоичного числа в восьмеричную систему счисления и обратно?
4) Как осуществляется перевод двоичного числа в шестнадцатеричную систему счисления и обратно?
5) Как получить двоичный эквивалент шестнадцатеричной цифры с помощью правила 8-4-2-1?

6. ПРАКТИЧЕСКАЯ РАБОТА 3. ТИПЫ ДАННЫХ ДЛЯ ПРЕДСТАВЛЕНИЯ ЦЕЛЫХ ЧИСЕЛ В ЭВМ. ВЫПОЛНЕНИЕ ОПЕРАЦИЙ НАД ЦЕЛЫМИ ЧИСЛАМИ. ЗАДАНИЯ 7 – 10

6.1. Содержание, варианты и примеры выполнения заданий

Задание 7. Представить десятичное отрицательное число X в машинном виде (формат short языка Java). Перевод в дополнительный код осуществить тремя способами. Варианты задания представлены в таблице 8.

Таблица 8 – Варианты для задания 7

НВ	Десятичное число X	НВ	Десятичное число X	НВ	Десятичное число X
1	− 2753	11	− 12457	21	− 892
2	− 3257	12	− 675	22	− 5945
3	− 10476	13	− 3579	23	− 9145
4	− 847	14	− 15480	24	− 25478
5	− 1089	15	− 18453	25	− 19245
6	− 27483	16	− 11900	26	− 599
7	− 9654	17	− 1356	27	− 29341
8	− 8346	18	− 30234	28	− 498
9	− 998	19	− 7982	29	− 2005
10	− 5467	20	− 8631	30	− 3985

Пример выполнения задания 7. $X= -156_{10}$.

Целое число в ЭВМ представляется n двоичными разрядами. Старший (самый левый) разряд – знаковый. У положительных чисел знаковый разряд равен 0, у отрицательных – 1. Число -156_{10} входит в диапазон формата short (от −32768 до 32767). Для числа в формате short n=16.

Java использует представление целых чисел в дополнительных кодах. Любой инверсный код (обратный или дополнительный) целого положительного числа равен его прямому коду.

Дополнительный код отрицательного получают как дополнение до границы диапазона (2^{n-1}). Это и есть первый способ получения дополнительного кода:

Знак=1,

Значение* = 2^{n-1} — |Значение|,

где* Значение* − *значение числа в дополнительном коде

1 способ. $2^{15} - 156 = 32768 - 156 = 32612$.

Переведем число 32612 в двоичную систему счисления. Для уменьшения числа делений, осуществим сначала перевод в восьмеричную систему, а затем, каждую восьмеричную цифру результата представим ее двоичным эквивалентом.

```
_ 3 2 6 1 2 | 8
  3 2 6 0 8 _ 4 0 7 6 | 8
            4   4 0 7 2 _ 5 0 9 | 8
                      4   5 0 4   6 3 | 8
                          5   5 6   7
                              7
```

Результат: **1111111101100100** (знаковый разряд равен 1, т.к. число отрицательное).

2 способ. *Обратный код отрицательного числа получают из прямого путем инвертирования (изменения на противоположные) значений всех его разрядов, кроме знакового. Дополнительный код отрицательного числа получают путем прибавления 1 к обратному коду.*

Получим сначала прямой код −156. Для этого переведем число 156 в двоичную систему счисления:

```
_ 1 5 6 | 8
  1 5 2 _ 1 9 | 8
        4   1 6   2
            3
```

Результат перевода: 010 011 100.

Прямой код (n=16): 10000000 10011100 (знаковый разряд равен 1).

Обратный код: 11111111 01100011 (инверсия всех кроме знакового).

\+ 1

———————————

Дополнительный код: **11111111 01100100** (к обратному коду прибавили 1)

Проверка (сумма прямого и дополнительного кода числа равна границе диапазона):

+	1	0	0	0	0	0	0	0	1	0	0	1	1	1	0	0	***Прямой код числа -156***
	1	1	1	1	1	1	1	1	0	1	1	0	0	1	0	0	***Дополнительный код -156***
	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	***2^{15} – граница диапазона***

3 способ. Получаем дополнительный код из прямого: справа налево все разряды до первой единицы включительно оставляем без изменения, знаковый разряд оставляем без изменения, остальные разряды инвертируем.

10000000 10011100 → **11111111 01100100** .

Результаты перевода в дополнительный код, полученные первым, вторым и третьим способами совпали. Таким образом, число -156_{10} в формате short , будет выглядеть так:

1	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**0**	**1**	**1**	**0**	**0**	**1**	**0**	**0**

Задание 8. Проимитировать действия АЛУ при выполнении фрагмента java-программы.

```
…
byte a = x, b = y;
int c = − a;
byte d = (byte)(− a);
int e = − b;
byte f = (byte)(− b);
…
```

Варианты задания представлены в таблице 9. значения переменных **x** и **y** приводятся в десятичной системе счисления.

Таблица 9 – Варианты для задания 8

НВ	x	y	НВ	x	y	НВ	x	y
1	54	−54	11	58	−58	21	55	−55
2	27	−27	12	25	−25	22	22	−22
3	38	−38	13	34	−34	23	31	−31
4	45	−45	14	48	−48	24	43	−43
5	91	−91	15	99	−99	25	96	−96
6	83	−83	16	87	−87	26	80	−80
7	75	−75	17	72	−72	27	79	−79
8	62	−62	18	69	−69	28	65	−65
9	111	−111	19	108	−108	29	119	−119
10	120	−120	20	115	−115	30	122	−122

Пример выполнения операции задания 8: x = 44; y = −44;

Операция «унарный минус» – это изменение знака числа (изменение знака числа): B = −A. Если A – положительное, то B – отрицательное и наоборот. Модули чисел A и B равны.

				byte a = 44; 00101100	44
byte int	00000000	00000000	00000000	00101100	44
int c = −a;	11111111	11111111	11111111	11010100	−44
				byte d = (byte)(−a); 11010100	−44

---

				byte b = −44; 11010100	−44
byte int	11111111	11111111	11111111	11010100	−44
int e = −b;	00000000	00000000	00000000	00101100	44
				byte f = (byte)(−b); 00101100	44

Задание 9. Проимитировать действия АЛУ при выполнении фрагмента java-программы.

```
…
int a = x, b = y;
int c = a − b;
…
```

Варианты задания представлены в таблице 10. значения переменных **x** и **y** приводятся в десятичной системе счисления. В каждом варианте нужно решить три примера и обосновать результаты.

Пояснение. В данном фрагменте java-программы определено три целочисленных переменных с именами a, b и c. Каждая из этих переменных занимает 4 байта в оперативной памяти (тип int*).*

*В Java рекомендуется выбирать 32-разрядный формат (**int**) для представления одиночных целых переменных, даже если их значения будут изменяться в диапазоне типа* byte*. Это обусловлено двумя причинами:*

во-первых, при выполнении действий над значениями коротких целых типов результат все равно всегда расширяется до int*:*

во-вторых, одиночные переменные и значения типа byte и short в Java могут на самом деле храниться в четырех байтах (для ускорения операций с ними), так что короткие типы характеризуют скорее «поведение», а не размер переменной (но если, например, создается массив из тысячи значений типа byte, то он будет занимать в памяти ровно 1000 байт).

АЛУ выполняет операции сложения и вычитания на двоичном сумматоре (двоичного вычитателя в АЛУ нет). Знаковые разряды слагаемых складываются на двоичном сумматоре, как и все остальные разряды. Т.к. слагаемые поступают на двоичный сумматор в дополнительных кодах, то и результат будет получен в дополнительном коде. Если результат положительный, то его дополнительный код равен прямому коду.

Таблица 10 – Варианты для задания 9

НВ	x	y	НВ	x	y	НВ	x	y
1	295	192	11	475	273	21	821	345
	−295	−192		−475	−273		−821	−345
	−20151	20456		−27643	11673		−28015	5094
2	348	256	12	749	531	22	1673	951
	−348	−256		−749	−531		−1673	−951
	20151	−20151		27643	−11673		28015	−5094
3	545	341	13	2639	1275	23	8531	5476
	−545	−341		−2639	−1275		−8531	−5476
	−15472	19345		−16345	17521		−4271	3005
4	948	721	14	3645	1368	24	5121	3114
	−948	−721		−3645	−1368		−5121	−3114
	15472	−19345		16345	−17521		4271	−3005

Продолжение таблицы 10

НВ	x	y	НВ	x	y	НВ	x	y
5	721	456	15	1629	985	25	898	769
	−721	−456		−1629	−985		−898	−769
	−21572	18300		−23476	14327		−10254	25700
6	458	276	16	2591	1983	26	2345	1234
	−458	−276		−2591	−1983		−2345	−1234
	21572	−18300		23476	−14327		10254	−25700
7	1503	642	17	539	295	27	3456	2457
	−1503	−642		−539	−295		−3456	−2457
	−15700	19365		−19340	16750		−12005	22900
8	1703	895	18	721	482	28	7541	5265
	−1703	−895		−721	−482		−7541	−5265
	15700	−19365		19340	−16750		12005	−22900
9	991	574	19	4973	3521	29	2356	1987
	−991	−574		−4973	−3521		−2356	−1987
	−29345	5700		−22956	12354		−25897	20431
10	875	628	20	3271	1654	30	971	492
	−875	−628		−3271	−1654		−971	−492
	29345	−5700		22956	−12354		25897	−19431

Пример выполнения задания 9.

1) x = −7590, y = −4385;
c = a – b = x – y = −7590 − (−4385) = −7591 + 4385= − 3205;

Получим прямые коды слагаемых в 32-битном формате.

```
_ 7 5 9 0 |8
  7 5 8 4 _ 9 4 8 |8
          6   9 4 4 _ 1 1 8 |8
              4   1 1 2 _ 1 4 |8
                  6     8   1
                        6
```

$(-7590)_{пр}$ = 10000000 00000000 00011101 10100110 ;
$(-7590)_{доп}$ = 11111111 11111111 11100010 01011010;

```
_ 4 3 8 5 |8
  4 3 8 4 _ 5 4 8 |8
          1   5 4 4 _ 6 8 |8
              4   6 4 _ 8 |8
                  4   8   1
                      0
```

$(4385)_{пр}$ = $(4385)_{доп}$ = 00000000 00000000 00010001 00100001

Выполнение операции на двоичном сумматоре АЛУ:

a	+	11111111	11111111	11100010	01011010	$(-7590)_{доп}$
b		00000000	00000000	00010001	00100001	$(4385)_{доп}$
c		11111111	11111111	11110011	01111011	$(-3205)_{доп}$
		10000000	00000000	00001100	10000101	$(-3205)_{пр}$

Проверка: $5\cdot 8^0 + 0\cdot 8^1 + 2\cdot 8^2 + 6\cdot 8^3 = 5 + 128 + 3072 = 3205$

2) x = 7590, y = 4385;
c = a – b = x – y = 7590 − 4385 = 7591 + (− 4385) = 3205;
Далее пример решается аналогично предыдущему.

3) x = 26592, y = −19301;
c = a – b = x – y = 26592 − (−19301) = 26592 + 19301 = 45893;
Далее пример решается аналогично предыдущим.

Задание 10. Проимитировать операцию сложения двоичных чисел с фиксированной точкой (тип int) на двоичном сумматоре. Объяснить знак результата, описать исключительную ситуацию. Перевести слагаемые из двоичной системы счисления в десятичную при помощи программы «Калькулятор» (<Вид> → <Программист>) и доказать, что сумма действительно выходит за границы диапазона int. Для отрицательных слагаемых предварительно получить их прямые коды (для этого использовать способ 3).

Варианты задания представлены в таблице 11.

Таблица 11 – Варианты для задания 10

НВ	Слагаемые
1	01101001 11000001 11000011 11000011, 00110001 00111111 00101110 00110010
2	11111100 00101010 11000011 11000111, 10000011 00111000 10100011 10100110
3	00001111 10110011 10011000 00001111, 01111010 01110111 00111100 10100111
4	10000001 10000011 00000000 11110011, 11111110 00111101 10100101 01011100
5	00110100 11110000 10100101 11001101, 01011101 00101111 00010001 10100111
6	11111111 11111111 01011100 10100001, 10000000 00000000 10011100 11111111
7	01000000 11110000 01011010 11101110, 00111111 00011000 01000000 01111110

Продолжение таблицы 11

НВ	Слагаемые
8	11111111 11111111 00000001 11001100, 10000000 00000000 01010101 00111101
9	01111111 11110011 01010101 11111111, 00000000 00011100 11111100 11110111
10	11111000 11001100 11100011 11001110, 10000111 00001100 11100010 11111100
11	01101001 11000001 11000011 11000011, 00110001 00111111 00101110 00110010
12	11111100 00101010 11000011 11000111, 10000011 00111000 10100011 10100110
13	00001111 10110011 10011000 00001111, 01111010 01110111 00111100 10100111
14	10000001 10000011 00000000 11110011, 11111110 00111101 10100101 01011100
15	00110100 11110000 10100101 11001101, 01011101 00101111 00010001 10100111
16	11111111 11111111 01011100 10100001, 10000000 00000000 10011100 11111111
17	01000000 11110000 01011010 11101110, 00111111 00011000 01000000 01111110
18	11111111 11111111 00000001 11001100, 10000000 00000000 01010101 00111101
19	01111111 11110011 01010101 11111111, 00000000 00011100 11111100 11110111
20	11111000 11001100 11100011 11001110, 10000111 00001100 11100010 11111100
21	01101001 11000001 11000011 11000011, 00110001 00111111 00101110 00110010
22	11111100 00101010 11000011 11000111, 10000011 00111000 10100011 10100110
23	00001111 10110011 10011000 00001111, 01111010 01110111 00111100 10100111
24	10000001 10000011 00000000 11110011, 11111110 00111101 10100101 01011100
25	00110100 11110000 10100101 11001101, 01011101 00101111 00010001 10100111
26	11111111 11111111 01011100 10100001, 10000000 00000000 10011100 11111111
27	01000000 11110000 01011010 11101110, 00111111 00011000 01000000 01111110

Продолжение таблицы 11

НВ	Слагаемые
28	11111111 11111111 00000001 11001100, 10000000 00000000 01010101 00111101
29	01111111 11110011 01010101 11111111, 00000000 00011100 11111100 11110111
30	11111000 11001100 11100011 11001110, 10000111 00001100 11100010 11111100

Слагаемые: 11111010 10001100 11101011 11001110,
10000101 01001100 11100010 11111100

На двоичном сумматоре АЛУ будет получен следующий результат:

$$\begin{array}{rl} + & \underline{1}0000101\ 01001100\ 11100010\ 11111100 \\ & \underline{1}1111010\ 10001100\ 11101011\ 11001110 \\ \hline & \underline{0}1111111\ 11011001\ 11001110\ 11001010 \end{array}$$

Знак суммы отличается от равных знаков слагаемых. Такая ситуация квалифицируется как переполнение с фиксированной точкой.

Переполнение – это исключительная ситуация, которая может возникнуть только при сложении чисел с одинаковыми знаками. Процессор, как правило, реагирует на подобную ситуацию установкой соответствующего бита в регистре флагов прерываний.

Проверка.

Первое слагаемое – *отрицательное*:

1111101 0 10001100 11101011 11001110 – дополнительный код,

1000010 1 01110011 00010100 00110010 – прямой код.

В десятичной системе счисления это число –91427890.

Примечание: для положительных чисел дополнительный код равен прямому.

Второе слагаемое – *отрицательное*:

1000010 1 01001100 11100010 11111100 – дополнительный код,

1111010 1 10110011 00011101 00000100 – прямой код.

В десятичной системе счисления это число –2058558724.

–91427890 + (–2058558724) = –2149986614.

Диапазон типа данных int:

от $-2^{31}=-2147483648$ *до* $2^{31}-1=2147483647$

Полученная сумма выходит за границы диапазона int, следовательно, она не поместится в 32-разрядной сетке.

6.2. Контрольные вопросы

1) Какой общий формат используется для представления целых чисел в ЭВМ?
2) Чем отличаются типы byte, short, int, long, используемые для представления целых чисел в Java.
3) Как определить границы диапазона того или иного типа, используемого для представления целых чисел? Приведите примеры.
4) Каким образом представляются в ЭВМ целые отрицательные числа?
5) Чему равен дополнительный код положительного числа.
6) Дайте определение дополнительного кода отрицательного числа и опишите три способа его получения.
7) Как из дополнительного кода отрицательного числа получить его прямой код?
8) Как имея машинное представление положительного числа получить машинное представление противоположного ему отрицательного числа? Получите прямой код числа 678. На основе этого представления получите дополнительный код числа −678 (используйте третий способ получения дополнительного кода).
9) Почему можно сказать, что все целые числа (отрицательные и неотрицательные) представляются Java в дополнительном коде?
10) Как выглядит машинное представление левой границы диапазона для типов byte, int?
11) Как выглядит машинное представление правой границы диапазона для типов byte, int?
12) Представьте правую границу диапазона типа long (максимальное положительное целое число, используемое в ЭВМ) в виде восьмеричного и шестнадцатеричного числа. По какому правилу вы осуществили перевод из машинного представления?
13) Как выглядят числа 0 и −1 в формате для любого целого типа?
14) Какие арифметические операции определены для целых чисел? 22) Как выполняется операция изменения знака числа?
15) Какой тип имеет результат арифметической операции над операндами типа byte и short?
16) На какой операции базируются операции вычитания, умножения и деления целых чисел(поясните)? Какой операционный элемент используется для исполнения этой операции?
17) Почему не нужно проектировать электрическую схему двоичного вычитателя? На каком операционном элементе и как осуществляется вычитание целых чисел?
18) Сформулируйте правила двоичного сложения с учетом переноса. Что происходит со знаковыми разрядами при сложении на двоичном сумматоре?
19) Слагаемые поступают на двоичный сумматор в дополнительных кодах. В каком коде будет получен результат? Если результат отрицательный, как узнать его модуль?
20) Какая исключительная ситуация может возникнуть при сложении (вычитании) чисел с фиксированной точкой? Приведите примеры.

7. ПРАКТИЧЕСКАЯ РАБОТА 4. ПРЕДСТАВЛЕНИЕ ВЕЩЕСТВЕННЫХ ЧИСЕЛ В ЭВМ. ТИПЫ ДАННЫХ С ПЛАВАЮЩЕЙ ТОЧКОЙ. ВЫПОЛНЕНИЕ ОПЕРАЦИЙ НАД ВЕЩЕСТВЕННЫМИ ЧИСЛАМИ. ЗАДАНИЯ 11 – 13

7.1. Содержание, варианты и примеры выполнения заданий

Задание 11. Представить десятичные числа в формате IEEE-754 (Single). Проверить результат путем обратных преобразований. Варианты задания представлены в таблице 12.

Таблица 12 – Варианты для задания 11

НВ	Число 1	Число 2	НВ	Число 1	Число 2
1	463.752	− 463.752	16	545.129	− 545.129
2	567.321	− 567.321	17	461.739	− 461.739
3	834.125	− 834.125	18	577.638	− 577.638
4	542.791	− 542.791	19	635.895	− 635.895
5	123.247	− 123.247	20	792.225	− 792.225
6	647.986	− 647.986	21	199.991	− 199.991
7	524.479	− 524.479	22	267.762	− 267.762
8	385.873	− 385.873	23	352.925	− 352.925
9	850.654	− 850.654	24	925.352	− 925.352
10	864.805	− 864.805	25	831.318	− 831.318
11	300.256	− 300.256	26	657.758	− 657.758
12	341.775	− 341.775	27	775.527	− 775.527
13	256.643	− 256.643	28	185.618	− 185.618
14	351.345	− 351.345	29	593.423	− 593.423
15	723.379	− 723.379	30	690.772	− 690.772

Пример выполнения задания 11. Числа: 326,527 и −326,527.

Переводим в двоичную систему (можно через восьмеричную) отдельно целую и дробную часть числа 326,527.

```
_ 3 2 6 | 8          |8
  3 2 0 _ 4 0        |5
        6   4 0
            0
```

```
  5   |  0  |  6
1 0 1 | 0 0 0 | 1 1 0
```

Чтобы представить мантиссу числа с точностью 24 бита (формат IEEE-754, Single), нужно определить 8 восьмеричных цифр. 3 цифры получилось в целой части, значит в дробной надо определить 5.

0,		5	2	7
	x			8
4		2	1	6
	x			8
1		7	2	8
	x			8
5		8	2	4
	x			8
6		5	9	2
	x			8
4		7	3	6

101000110.100001101110100 =

$=1.01000110100001101110100 \cdot 2^8$

Получили мантиссу в нормализованном виде (число 326,527 входит в диапазон нормализованных чисел формата IEEE-754, Single) и порядок (8_{10}).

Экспонента (сдвинутый порядок):
$8+127=135_{10}=207_8=10000111_2$

_	1	3	5	|	8														
	1	2	8	_	1	6	|	8			2			0			7		
			7		1	6		2		0	**1**	**0**	**0**	**0**	**0**	**1**	**1**	**1**
						0													

Представление числа 326.527_{10} в формате формат IEEE-754-Single (float – Java): **0 10000111 01000110100001101110100** (знак, смещенный порядок и остаток от мантиссы).

Проверка.

Формула вычисления значения десятичного числа, из значений, представленных в полях формата IEEE-754 (для нормализованных чисел):

$$F = (-1)^s \cdot 2^{(E-2^{(b-1)}+1)} \cdot (1 + M / 2^n)$$

При вычислении значения числа используется истинный, а не смещенный порядок $E-(2^{b-1}-1)$, к остатку от мантиссы прибавляется целая часть мантиссы, равная 1. Деление остатка от мантиссы M на 2^n обозначает отрицательные степени двойки, соответствующие весам разрядов M, например, $2^{-1}=1/2$.

$F=(-1)^0 2^{(135-127)} (1+(2^{21}+2^{17}+2^{16}+2^{14}+2^9+2^8+2^6+2^5+2^4+2^2) /2^{23})=$

$=2^8(1+1/2^2+1/2^6+1/2^7+1/2^9+1/2^{14}+1/2^{15}+1/2^{17}+1/2^{18}+1/2^{19}+1/2^{21})=$

$=2^8+2^6+2^2+2+1/2+1/2^6+1/2^7+1/2^9+1/2^{10}+1/2^{11}+1/2^{13} =$

$=326+ (4096+128+64+16+8+4+2+1)/8192=326+4319/8192\approx$

$\approx 326.527_{10}$

Представление числа -326.527_{10} в формате float:

1 10000111 01000110100001101110100 (отличается от представления числа 326.527_{10} только значением знакового разряда).

Задание 12. Промоделировать алгоритм сложения (вычитания) чисел с плавающей точкой C = A + B (C = A − B). Варианты задания представлены в таблице 13.

Таблица 13 – Варианты для задания 12

НВ	Числа A и B в формате с плавающей точкой	Операция
1	1 10000011 00010101110…0 0 01111110 10101000000…0	+
2	1 10000011 00010101110…0 0 01111110 10101000000…0	−
3	0 11101111 11101000000…0 1 11110011 01011110010…0	+
4	0 11101111 11101000000…0 1 11110011 01011110010…0	−
5	1 11110110 00101011100…0 0 11110010 11100100000…0	+
6	1 11110110 00101011100…0 0 11110010 11100100000…0	−
7	0 00010000 10001100000…0 1 00001100 01011011010…0	+
8	0 00010000 10001100000…0 1 00001100 01011011010…0	−
9	1 00011001 01111100000…0 1 00010111 00101110010…0	+
10	1 00011001 01111100000…0 1 00010111 00101110010…0	−
11	0 11100111 10110100000…0 1 11101010 00100111010…0	+
12	0 11100111 10110100000…0 1 11101010 00100111010…0	−
13	1 11001010 01011100100…0 1 11000101 01101000000…0	+
14	1 11001010 01011100100…0 1 11000101 01101000000…0	−
15	0 00010011 10010010010…0 1 00001110 00011100000…0	+
16	0 00010011 10010010010…0 1 00001110 00011100000…0	−
17	1 10000011 00010101110…0 0 01111110 10101000000…0	+
18	1 10000011 00010101110…0 0 01111110 10101000000…0	−

Продолжение таблицы 13

НВ	Числа А и В в формате с плавающей точкой	Операция
19	0 11101111 11101000000…0 1 11110011 01011110010…0	+
20	0 11101111 11101000000…0 1 11110011 01011110010…0	−
21	1 11110110 00101011100…0 0 11110010 11100100000…0	+
22	1 11110110 00101011100…0 0 11110010 11100100000…0	−
23	0 00010000 10001100000…0 1 00001100 01011011010…0	+
24	0 00010000 10001100000…0 1 00001100 01011011010…0	−
25	1 00011001 01111100000…0 1 00010111 00101110010…0	+
26	1 00011001 01111100000…0 1 00010111 00101110010…0	−
27	0 11100111 10110100000…0 1 11101010 00100111010…0	+
28	0 11100111 10110100000…0 1 11101010 00100111010…0	−
29	1 11001010 01011100100…0 1 11000101 01101000000…0	+
30	1 11001010 01011100100…0 1 11000101 01101000000…0	−

Для выполнения данного задания необходимо знать алгоритмы и правила, приведенные ниже.

Алгоритм сложения (вычитания) чисел, представленных в формате IEEE754.

1) Для каждого из слагаемых: если слагаемое принадлежит диапазону нормализованных чисел, мантисса слагаемого восстанавливается (учитывается единица целой части, которая не хранится в формате IEEE754).

2) *Вычисляется разность экспонент на сумматоре порядков(СП) по правилу вычитания беззнаковых чисел (подробно описано ниже).*

$$E_A - E_B = E_A + (-E_B)_{доп}$$

Если разность экспонент больше или равна числу разрядов мантиссы, то сумма принимается равной числу с большей экспонентой и выполнение алгоритма заканчивается.

ВЫРАВНИВАНИЕ ПОРЯДКОВ (п. 3, 4)

3) Экспонента результата принимается равной большей из экспонент слагаемых.

4) Мантисса числа с меньшей экспонентой (если разность экспонент отрицательная, то это число А, если разность экспонент положительная, то это число В) сдвигается вправо на число разрядов, равное модулю разности экспонент слагаемых (прямому коду разности экспонент).

5) Мантисса суммы (разности) чисел А и В вычисляется путем сложения (вычитания) мантисс слагаемых с помощью двоичного сумматора мантисс (СМ) по правилам сложения чисел со знаком, представленных в дополнительных кодах с удвоенным знаковым разрядом (подробно описаны ниже).

6) *Результат нормализуется по описанным ниже правилам.*

7) Если результат сложения (вычитания) мантисс отрицательный (знаковые разряды сумматора мантисс равны 1), то он преобразуется из дополнительного кода в прямой.

8) В поле S результата С записывается знак мантиссы (он же знак числа). В поле Е результата С записывается экспонента результата, полученная после выполнения процедуры нормализации. Если экспонента не равна нулю, то в поле М результата С записывается остаток от мантиссы (мантисса без старшей единицы), в противном случае – вся мантисса (со старшей единицей).

Правило вычитания беззнаковых (положительных) чисел:

$$E_A - E_B = E_A + (-E_B)_{доп}$$

Уменьшаемое поступает на сумматор порядков в прямом коде. Вычитаемое должно быть преобразовано в дополнительный код. Дополнительный код для беззнакового числа получают по тому же правилу, что и для отрицательного числа со знаком (за исключением требований для знакового разряда, т.к. его нет).

Если при вычислении разности беззнаковых (положительных) чисел нет переноса из старшего разряда сумматора, то разность порядков будет отрицательной и представленной в дополнительном коде. Если перенос есть, то разность порядков будет положительной и представленной в прямом коде.

Правило сложения чисел со знаком, представленных в дополнительном модифицированном (с удвоенным знаковым разрядом) коде.

Это правило основано на равенстве:

$$M_A - M_B = M_A + (-M_B)$$

Если числа поступают на двоичный сумматор в дополнительном коде, то и результат будет получен в дополнительном коде.

Вспомним, что дополнительный код положительного числа равен его прямому коду. Для получения дополнительного кода отрицательного числа, можно использовать одно из трех правил, рассмотренных в задании 7 (рекомендуется использовать третье правило). У модифицированного кода знаковый разряд удваивается.

Удвоение знакового разряда используется для упрощения проверки на возникновение ситуации переполнения: если при сложении чисел, представленных в дополнительных модифицированных кодах получен результат с разными знаковыми разрядами, значит, произошло переполнение (информационный разряд перекрыл первый (справа) знаковый).

Виды нарушения нормализации результата:

1) Нарушение нормализации влево наблюдается, когда при сложении (вычитании) мантисс произошло переполнение (знаковые разряды сумматора мантисс после выполнения операции – разные).

Корректируется сдвигом мантиссы вправо на 1 разряд (уменьшением в 2 раза) с соответствующим увеличением экспоненты на 1.

2) Нарушение нормализации вправо. Признаки: после выполнения операции знаковые разряды сумматора мантисс равны (переполнения не произошло), старший информационный разряд сумматора мантисс равен знаковому разряду, результат не равен $-\mathbf{2}^{-\mathbf{1}} = -\frac{\mathbf{1}}{\mathbf{2}}$ (у дополнительного кода этого числа знаковый и старший информационный разряды равны, тем не менее, это число считается нормализованным).

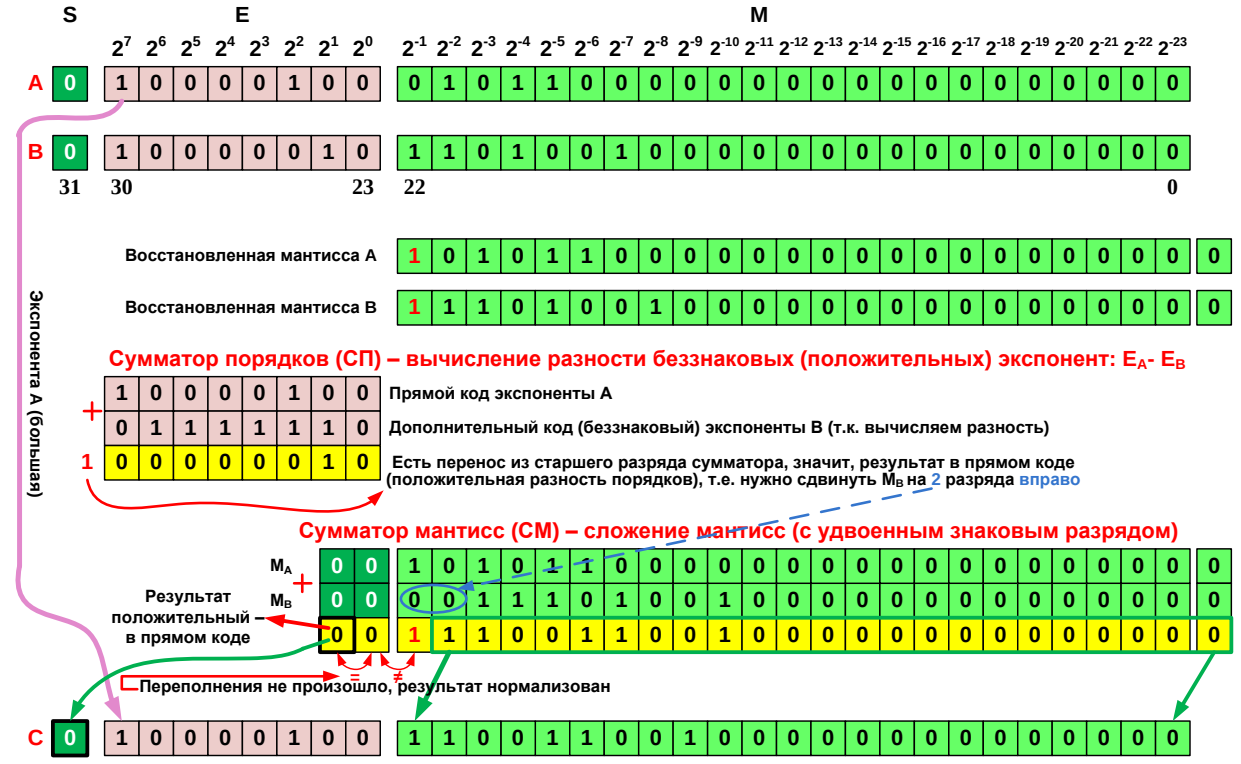
Корректируется последовательными сдвигами мантиссы влево (увеличение в 2 раза) до тех пор, пока знаковый разряд и старший информационный не станут разными. При каждом сдвиге экспонента уменьшается на 1, чтобы значение нормализуемого числа не изменилось.

Внимание! При нормализации результата сложения чисел с плавающей точкой (увеличении или уменьшении порядков) мы можем получить значение, выходящее из диапазона представления вещественных чисел, т.е. $\pm$ Infinity (бесконечность). Тот же результат можно получить и при выполнении операций умножения и деления вещественных чисел, т.к. порядки чисел при выполнении этих операций будут складываться или вычитаться.

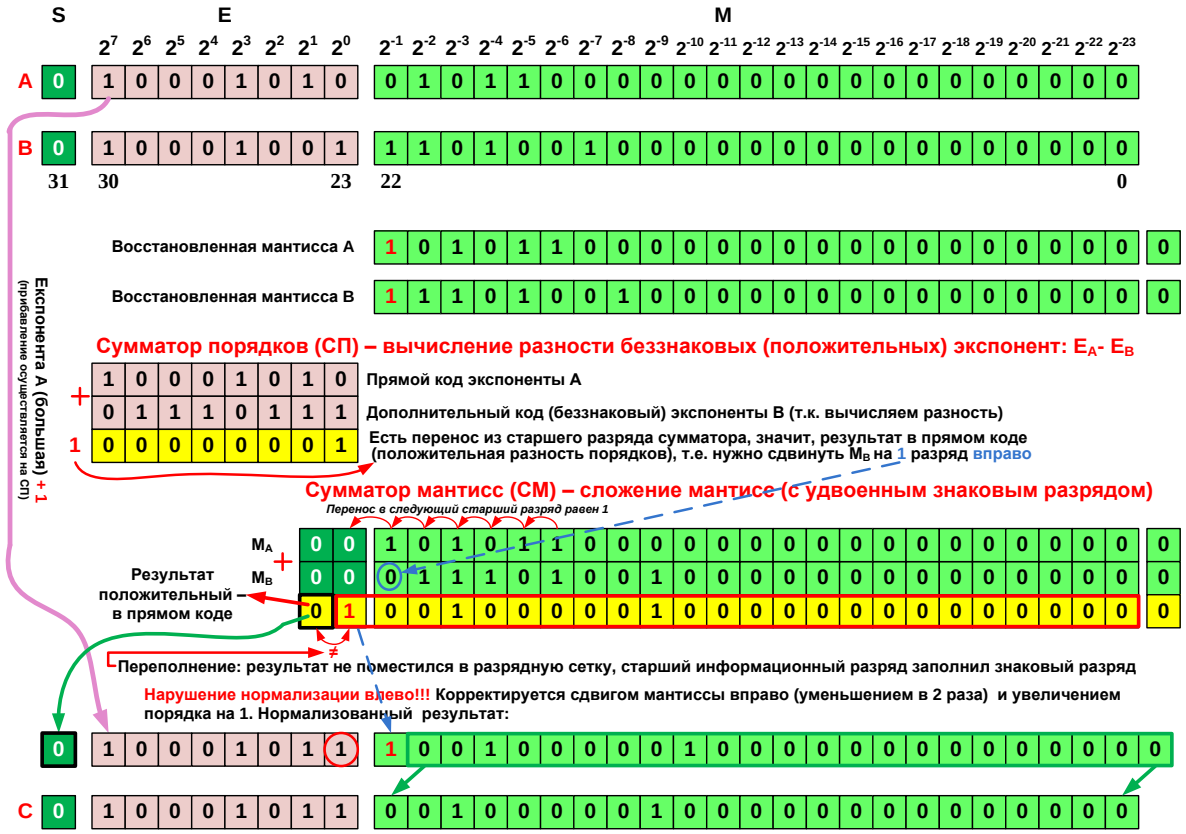
Примеры выполнения задания 12 приведены ниже.

Пример 12.1.

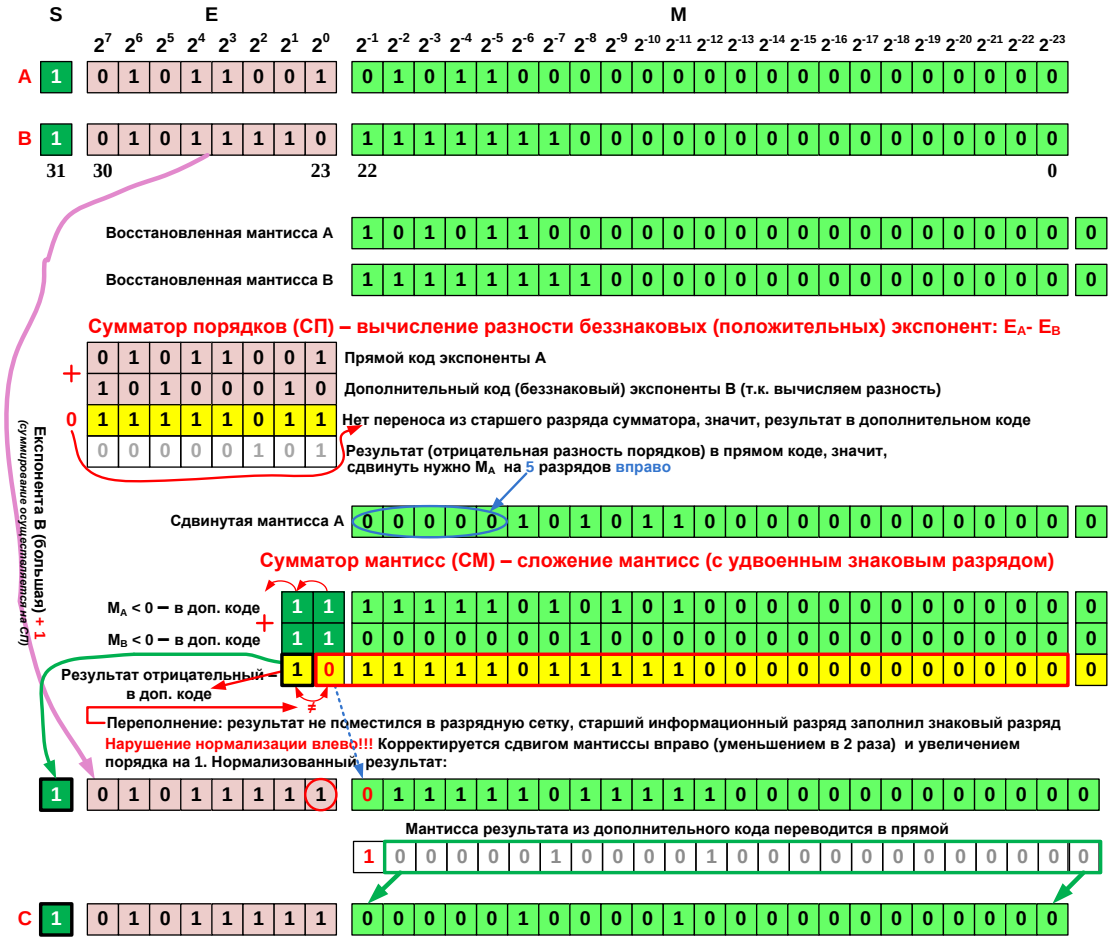
Пример 12.2.



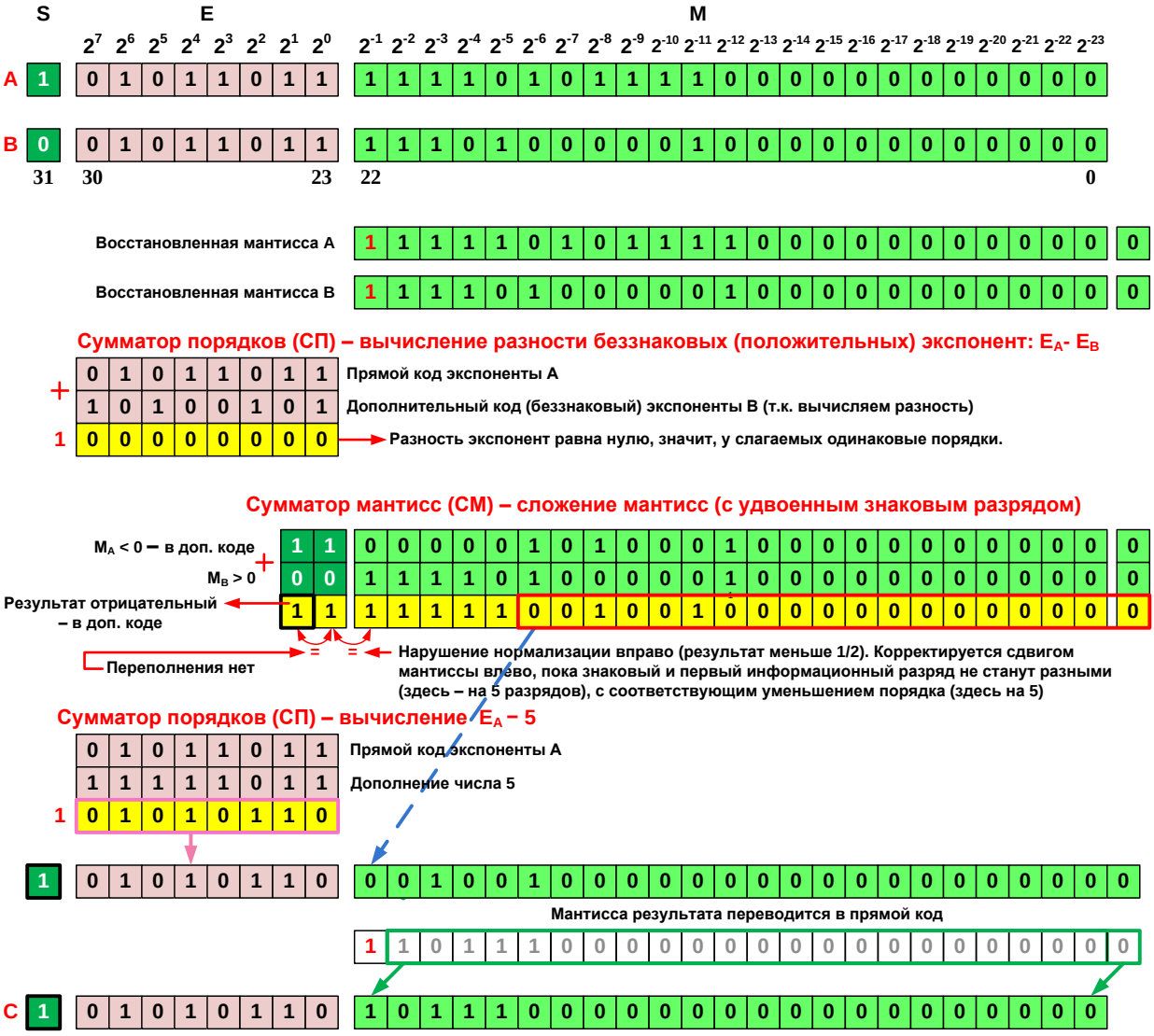
Пример 12.3.



Пример 12.4.

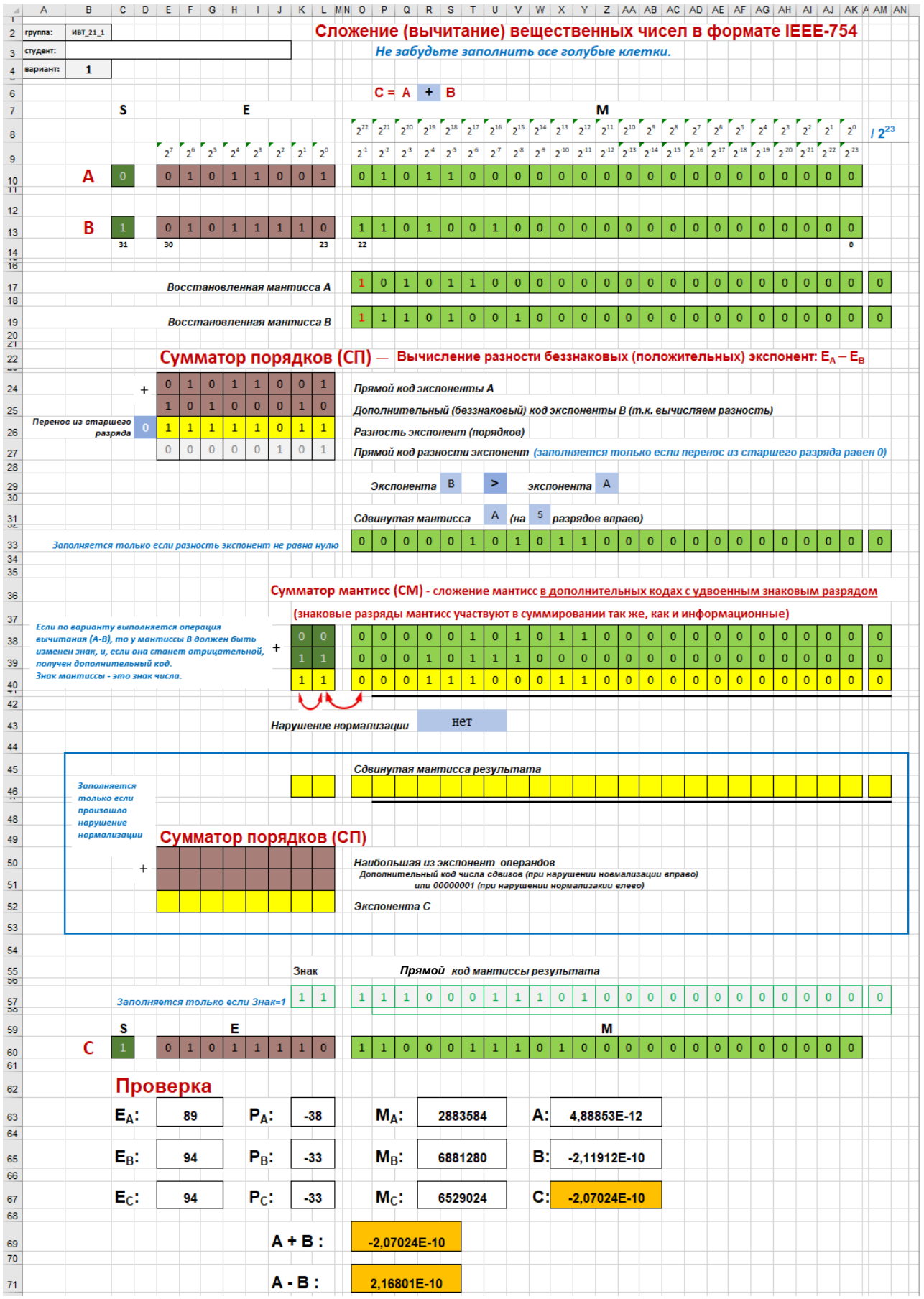


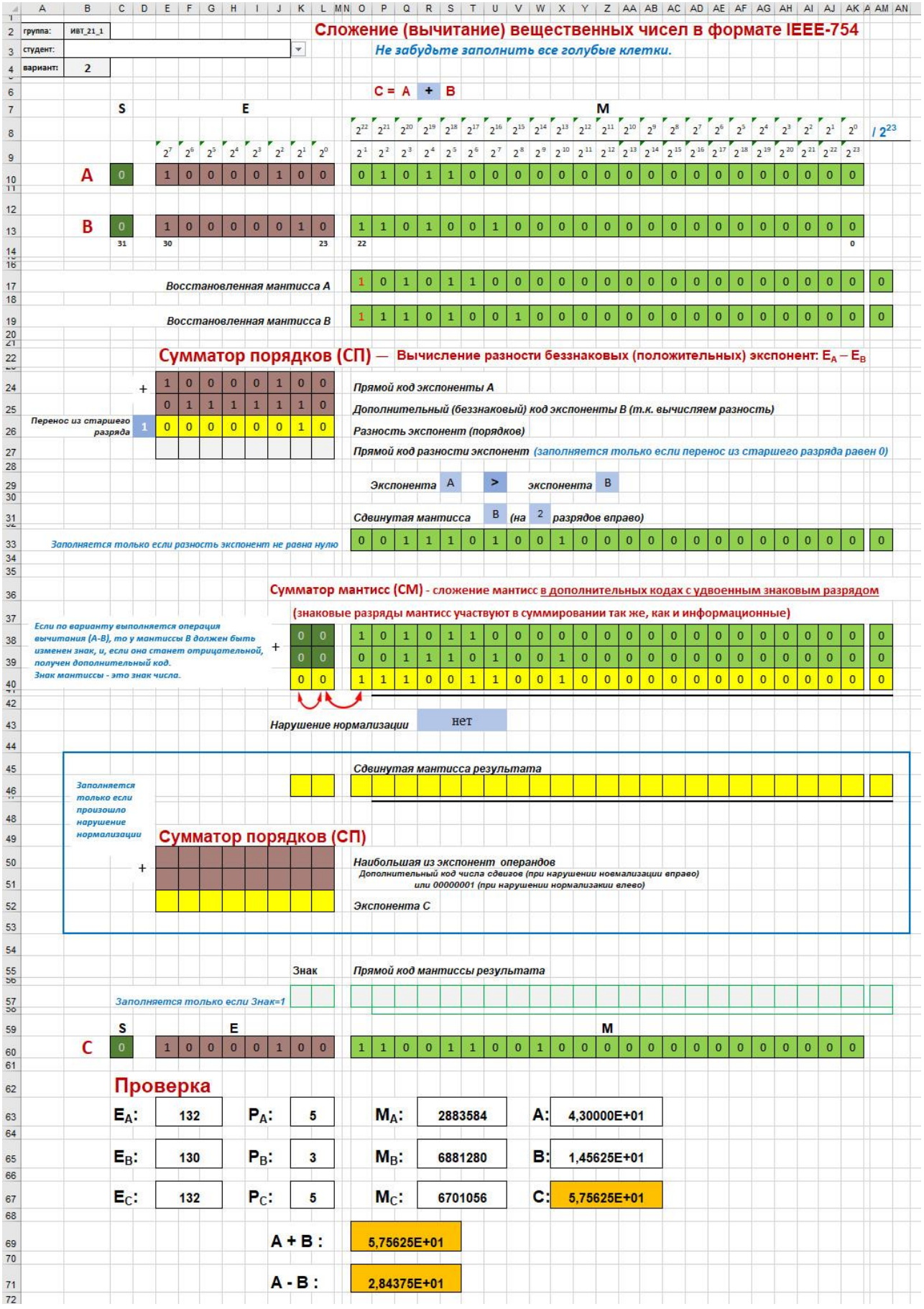
Пример 12.5.



Внимание! Задание 12 выполняется с помощью табличного процессора Excel. Шаблон соответствующего файла находится в дистанционном курсе. Заполните шаблон и загрузите файл в соответствующее электронное задание дистанционного курса. Ниже приведены варианты заполнения шаблонов для рассмотренных примеров реализации алгоритма.

Обратите внимание, что в шаблоне автоматически делается проверка результата: значения в ячейках «C:» и «A + B:» («A – B:») должны совпадать.


группа: ИВТ_21_1
студент:
вариант: 1
Сложение (вычитание) вещественных чисел в формате IEEE-754
Не забудьте заполнить все голубые клетки.
C = A + B
S
E
M
A
B
Восстановленная мантисса A
Восстановленная мантисса B
Сумматор порядков (СП) — Вычисление разности беззнаковых (положительных) экспонент: $E_A - E_B$
Прямой код экспоненты A
Дополнительный (беззнаковый) код экспоненты B (т.к. вычисляем разность)
Перенос из старшего разряда
Разность экспонент (порядков)
Прямой код разности экспонент (заполняется только если перенос из старшего разряда равен 0)
Экспонента B > экспонента A
Сдвинутая мантисса A (на 5 разрядов вправо)
Заполняется только если разность экспонент не равна нулю
Сумматор мантисс (СМ) - сложение мантисс в дополнительных кодах с удвоенным знаковым разрядом
(знаковые разряды мантисс участвуют в суммировании так же, как и информационные)
Если по варианту выполняется операция вычитания (A-B), то у мантиссы B должен быть изменен знак, и, если она станет отрицательной, получен дополнительный код.
Знак мантиссы - это знак числа.
Нарушение нормализации
нет
Сдвинутая мантисса результата
Заполняется только если произошло нарушение нормализации
Сумматор порядков (СП)
Наибольшая из экспонент операндов
Дополнительный код числа сдвигов (при нарушении нормализации вправо) или 00000001 (при нарушении нормализации влево)
Экспонента C
Знак
Прямой код мантиссы результата
Заполняется только если Знак=1
S
E
M
C
Проверка
E_A: 89
P_A: -38
M_A: 2883584
A: 4,88853E-12
E_B: 94
P_B: -33
M_B: 6881280
B: -2,11912E-10
E_C: 94
P_C: -33
M_C: 6529024
C: -2,07024E-10
A + B : -2,07024E-10
A - B : 2,16801E-10


группа: ИВТ_21_1
студент:
вариант: 2
Сложение (вычитание) вещественных чисел в формате IEEE-754
Не забудьте заполнить все голубые клетки.
C = A + B
S
E
M
A
B
Восстановленная мантисса A
Восстановленная мантисса B
Сумматор порядков (СП) — Вычисление разности беззнаковых (положительных) экспонент: $E_A - E_B$
Прямой код экспоненты A
Дополнительный (беззнаковый) код экспоненты B (т.к. вычисляем разность)
Перенос из старшего разряда
Разность экспонент (порядков)
Прямой код разности экспонент (заполняется только если перенос из старшего разряда равен 0)
Экспонента A > экспонента B
Сдвинутая мантисса B (на 2 разрядов вправо)
Заполняется только если разность экспонент не равна нулю
Сумматор мантисс (СМ) - сложение мантисс в дополнительных кодах с удвоенным знаковым разрядом
(знаковые разряды мантисс участвуют в суммировании так же, как и информационные)
Если по варианту выполняется операция вычитания (A-B), то у мантиссы B должен быть изменен знак, и, если она станет отрицательной, получен дополнительный код.
Знак мантиссы - это знак числа.
Нарушение нормализации
нет
Заполняется только если произошло нарушение нормализации
Сдвинутая мантисса результата
Сумматор порядков (СП)
Наибольшая из экспонент операндов
Дополнительный код числа сдвигов (при нарушении нормализации вправо) или 00000001 (при нарушении нормализакии влево)
Экспонента C
Знак
Прямой код мантиссы результата
Заполняется только если Знак=1
C
Проверка
E_A: 132 P_A: 5 M_A: 2883584 A: 4,30000E+01
E_B: 130 P_B: 3 M_B: 6881280 B: 1,45625E+01
E_C: 132 P_C: 5 M_C: 6701056 C: 5,75625E+01
A + B : 5,75625E+01
A - B : 2,84375E+01

группа:	ИВТ_21_1
студент:	
вариант:	3

Сложение (вычитание) вещественных чисел в формате IEEE-754

Не забудьте заполнить все голубые клетки.

C = A + B

	S	E	M
A	0	1 0 0 0 1 0 1 0	0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
B	0	1 0 0 0 1 0 0 1	1 1 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
	31	30 … 23	22 … 0

Восстановленная мантисса A: 1 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Восстановленная мантисса B: 1 1 1 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Сумматор порядков (СП) — Вычисление разности беззнаковых (положительных) экспонент: $E_A - E_B$

+	1 0 0 0 1 0 1 0		*Прямой код экспоненты A*
	0 1 1 1 0 1 1 1		*Дополнительный (беззнаковый) код экспоненты B (т.к. вычисляем разность)*
Перенос из старшего разряда	1	0 0 0 0 0 0 0 1	*Разность экспонент (порядков)*
			Прямой код разности экспонент (заполняется только если перенос из старшего разряда равен 0)

Экспонента A **>** *экспонента* B

Сдвинутая мантисса B *(на* 1 *разрядов вправо)*

Заполняется только если разность экспонент не равна нулю: 0 1 1 1 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Сумматор мантисс (СМ) - сложение мантисс в дополнительных кодах с удвоенным знаковым разрядом

(знаковые разряды мантисс участвуют в суммировании так же, как и информационные)

Если по варианту выполняется операция вычитания (A-B), то у мантиссы B должен быть изменен знак, и, если она станет отрицательной, получен дополнительный код.
Знак мантиссы - это знак числа.

+	0 0	1 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
	0 0	0 1 1 1 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
	0 1	0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Нарушение нормализации влево

Заполняется только если произошло нарушение нормализации

Сдвинутая мантисса результата: 0 0 | 1 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Сумматор порядков (СП)

+	1 0 0 0 1 0 1 0	*Наибольшая из экспонент операндов*
	0 0 0 0 0 0 0 1	*Дополнительный код числа сдвигов (при нарушении нормализации вправо) или 00000001 (при нарушении нормализации влево)*
	1 0 0 0 1 0 1 1	*Экспонента C*

Знак | *Прямой код мантиссы результата*

Заполняется только если Знак=1

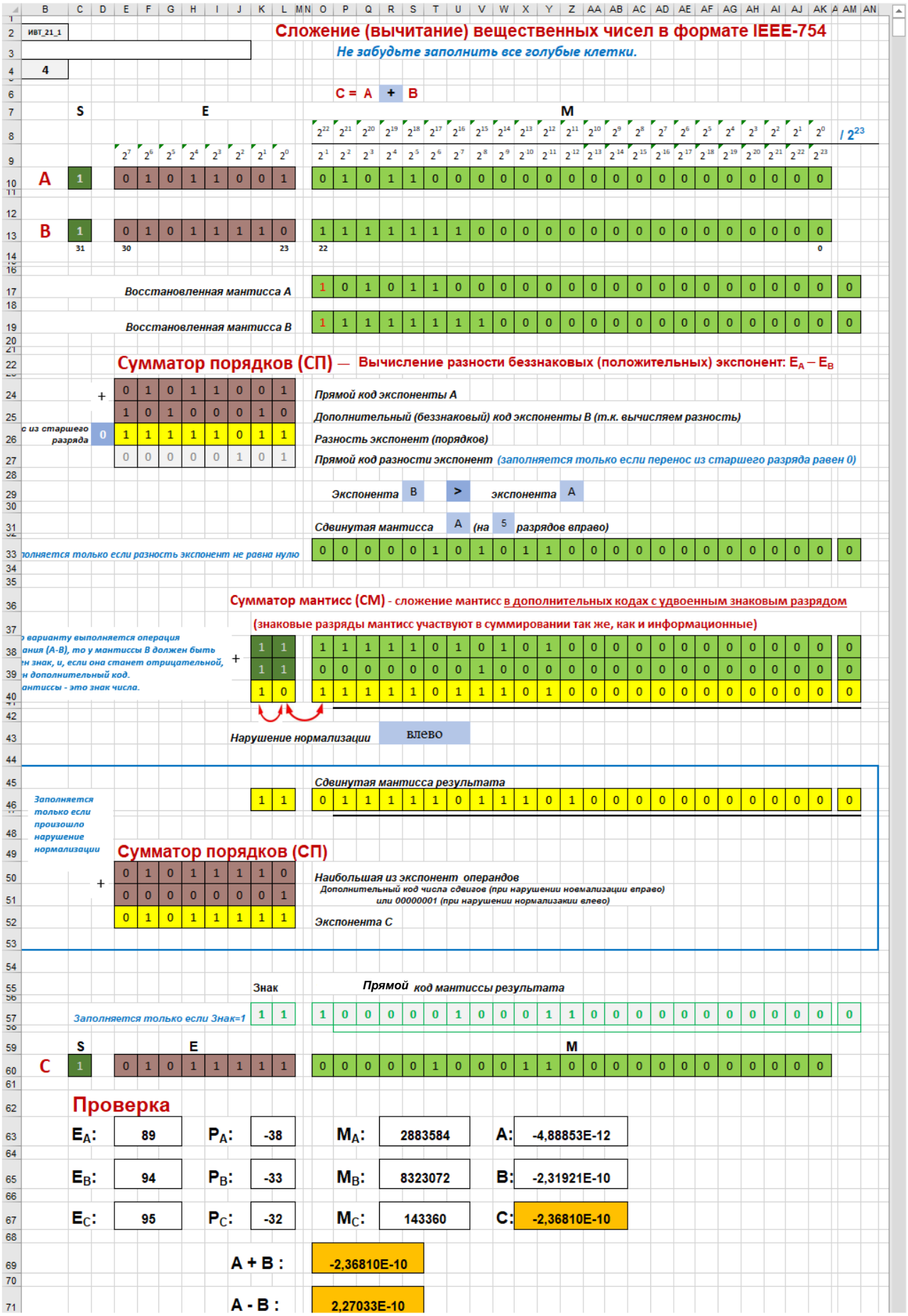
	S	E	M
C	0	1 0 0 0 1 0 1 1	0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

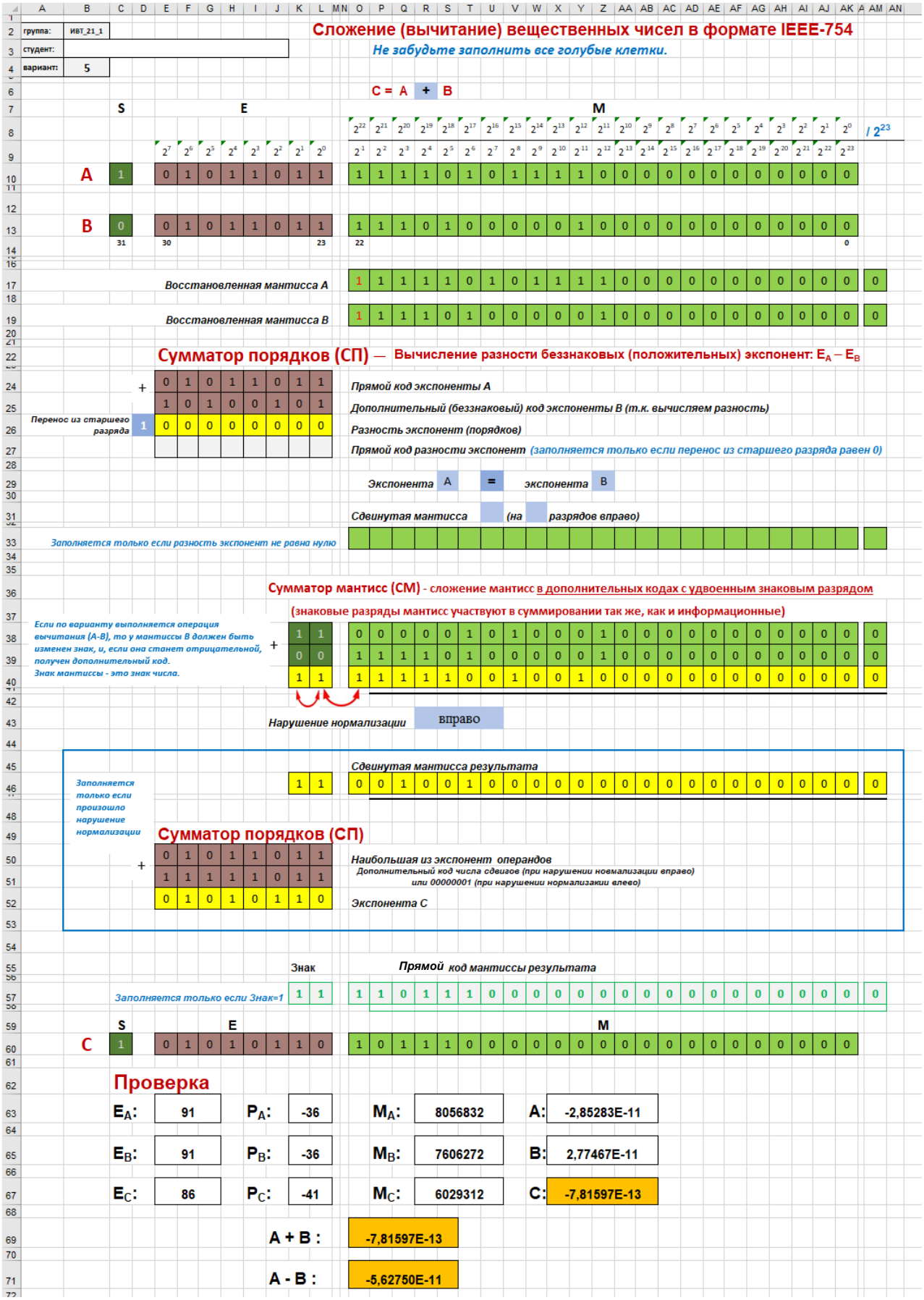
Проверка

E_A:	138	P_A:	11	M_A:	2883584	A:	2,75200E+03
E_B:	137	P_B:	10	M_B:	6881280	B:	1,86400E+03
E_C:	139	P_C:	12	M_C:	1064960	C:	4,61600E+03

A + B : 4,61600E+03

A - B : 8,88000E+02


ИВТ_21_1
4
Сложение (вычитание) вещественных чисел в формате IEEE-754
Не забудьте заполнить все голубые клетки.
C = A + B
S
E
M
A
B
Восстановленная мантисса A
Восстановленная мантисса B
Сумматор порядков (СП) — Вычисление разности беззнаковых (положительных) экспонент: E_A – E_B
Прямой код экспоненты A
Дополнительный (беззнаковый) код экспоненты B (т.к. вычисляем разность)
Разность экспонент (порядков)
Прямой код разности экспонент (заполняется только если перенос из старшего разряда равен 0)
Экспонента B > экспонента A
Сдвинутая мантисса A (на 5 разрядов вправо)
Сумматор мантисс (СМ) - сложение мантисс в дополнительных кодах с удвоенным знаковым разрядом
(знаковые разряды мантисс участвуют в суммировании так же, как и информационные)
Нарушение нормализации влево
Сдвинутая мантисса результата
Заполняется только если произошло нарушение нормализации
Сумматор порядков (СП)
Наибольшая из экспонент операндов
Дополнительный код числа сдвигов (при нарушении нормализации вправо) или 00000001 (при нарушении нормализации влево)
Экспонента C
Знак
Прямой код мантиссы результата
Заполняется только если Знак=1
Проверка
E_A: 89 P_A: -38 M_A: 2883584 A: -4,88853E-12
E_B: 94 P_B: -33 M_B: 8323072 B: -2,31921E-10
E_C: 95 P_C: -32 M_C: 143360 C: -2,36810E-10
A + B : -2,36810E-10
A - B : 2,27033E-10



Задание 13. После выполнения арифметической операции над двумя вещественными числами, представленными в формате IEEE754 – Single, получен некоторый результат. Вычислить десятичное значение и максимальные

(абсолютную и относительную) ошибки (погрешности представления) этого результата. Задание выполнить для трех чисел, предусмотренных вариантом. Ответы оформить в виде таблицы (как показано в примере выполнения задания) и проанализировать. Варианты задания 13 приведены в таблице 14.

Таблица 14 – Варианты для задания 13

НВ	Результат операции	НВ	Результат операции
1	0 00000000 11010000000000000000010	16	0 00000000 00010000000000000000011
	0 00000010 11010000000000000000010		0 00000010 00010000000000000000011
	1 11111010 11010000000000000000010		1 11111010 00010000000000000000011
2	0 00000000 01101000000000000000001	17	0 00000000 00011000000000000000101
	0 00000011 01101000000000000000001		0 00000100 00011000000000000000101
	1 11110011 01101000000000000000001		1 11110100 00011000000000000000101
3	1 00000000 11010000000000000000011	18	1 00000000 00010000000000000000010
	1 00000100 11010000000000000000011		1 00000101 00010000000000000000010
	0 11110100 11010000000000000000011		0 11100101 00010000000000000000010
4	1 00000000 01101000000000000000101	19	1 00000000 00011000000000000000110
	1 00000101 01101000000000000000101		1 00000110 00011000000000000000110
	0 11111101 01101000000000000000101		0 11110110 00011000000000000000110
5	0 00000000 00001000000000000001001	20	1 00000000 00111000000000000001001
	0 00000010 00001000000000000001001		1 00000111 00111000000000000001001
	1 11111010 00001000000000000001001		0 11100111 00111000000000000001001
6	0 00000000 00010000000000000000110	21	0 00000000 11010000011000000000010
	0 00000101 00010000000000000000110		0 00000101 11010000011000000000010
	1 11111101 00010000000000000000110		1 11110101 11010000011000000000010
7	0 00000000 00011000000000000000011	22	0 00000000 01101000100100000000001
	0 00000100 00011000000000000000011		0 00000110 01101000100100000000001
	0 11110100 00011000000000000000011		1 11111110 01101000100100000000001
8	1 00000000 00010000000000000001001	23	1 00000000 11010000110000000000011
	1 00000011 00010000000000000001001		1 00000111 11010000110000000000011
	0 11110011 00010000000000000001001		0 11100111 11010000110000000000011
9	1 00000000 00011000000000000000010	24	1 00000000 01101000001100100000101
	1 00000110 00011000000000000000010		1 00000010 01101000001100100000101
	0 11100110 00011000000000000000010		0 11110010 01101000001100100000101
10	1 00000000 10111000000000000000101	25	0 00000000 00001000100100000001001
	1 00000100 10111000000000000000101		0 00000011 00001000100100000001001
	0 11110100 10111000000000000000101		1 11110011 00001000100100000001001
11	0 00000000 01010000000000000001101	26	0 00000000 00010001101000000000110
	0 00000011 01010000000000000001101		0 00000100 00010001101000000000110
	1 11111011 01010000000000000001101		1 11110100 00010001101000000000110
12	0 00000000 11101000000000000001010	27	0 00000000 00011000101000000000011
	0 00000010 11101000000000000001010		0 00000101 00011000101000000000011
	1 11111010 11101000000000000001010		1 11100101 00011000101000000000011
13	1 00000000 11010000000000000000001	28	1 00000000 00010001011000000001001
	1 00000101 11010000000000000000001		1 00000110 00010001011000000001001
	0 11110101 11010000000000000000001		0 11010110 00010001011000000001001
14	1 00000000 01101000000000000000100	29	1 00000000 00011001001000010000010
	1 00000110 01101000000000000000100		1 00000111 00011001001000010000010
	0 11100110 01101000000000000000100		0 11100111 00011001001000010000010
15	0 00000000 00001000000000000010001	30	1 00000000 10111000101001001000101
	0 00000100 00001000000000000010001		1 00000100 10111000101001001000101
	0 11110100 00001000000000000010001		1 11011100 10111000101001001000101

Десятичные эквиваленты значений, представленных в поле M, используемые для подстановки в формулы для вычисления относительной погрешности, разрешается получить с помощью программы «Калькулятор» (Вид → Программист).

Пример выполнения задания 13.

1) Представление первого результата операции в формате IEEE754 – Single:
0 00000000 10101010100000111100011.

Экспонента числа равна нулю, следовательно это число входит в диапазон денормализованных (положительных) чисел формата IEEE754 – Single.
Десятичное значение денормализованного числа (Single) вычисляется по формуле:

$$F = (-1)^S \cdot 2^{(E-(2^{(8-1)}-1)+1)} \cdot M / 2^{23} = (-1)^S \cdot 2^{(E-126)} \cdot M / 2^{23}$$

У денормализованного числа в формате IEEE754 экспонента E=0.

$$F = (-1)^0 \cdot 2^{-126} \cdot M / 2^{23} = 5587427 \cdot 2^{-149} \approx 5587427 \cdot 1{,}4 \cdot 10^{-45} = 7{,}822398\,e-39$$

Абсолютная погрешность равна половине шага чисел рассматриваемого диапазона.
Шаг денормализованных чисел: $h=2^{-149}$. Соответственно максимальная абсолютная ошибка δ будет равна 1/2 шага числа: $\delta=2^{-150}=7{,}006492e\text{-}46$.
Максимальная относительная ошибка для денормализованного (E=0) числа (single/double):

$$\xi = \frac{2^{(E-150)}}{2^{(E-126)} \cdot \frac{M}{2^{23}}} \cdot 100\% = \frac{2^{-150} \cdot 2^{23}}{2^{-126} \cdot M} \cdot 100\% = \frac{1}{2M} \cdot 100\% =$$

$$= \frac{1}{2 \cdot 5587427} \cdot 100\% = 8{,}94866e-6\ (\%).$$

2) Представление второго результата операции в формате IEEE754 – Single:
0 00000110 10101010100000111100011.
Экспонента числа равна 6 (больше нуля и меньше 255_{10} (11111111_2)) , следовательно, это число входит в диапазон нормализованных (положительных) чисел формата IEEE754 – Single.
Десятичное значение нормализованного числа (Single) вычисляется по формуле:

$$F = (-1)^S \cdot 2^{(E-127)} \cdot (1 + M / 2^{23})$$

$$F = (-1)^0 \cdot 2^{(6-127)} \cdot (1 + M / 2^{23}) = 2^{-121} \cdot (1 + 5587427 / 8388608) \approx$$

$$\approx 3{,}761582 \cdot 10^{-37} \cdot 1{,}666073 \approx 6{,}267070\,e-37$$

Шаг нормализованных чисел формата IEEE754 – Single: $h=2^{(E-150)}$.
Максимальная абсолютная погрешность представления числа:
$\delta=h/2=2^{(E-151)}=2^{(6-151)}=2^{-145}=2{,}242077e-44.$

Максимальная относительная ошибка (погрешность представления) нормализованного числа (Single):

$$\xi=\frac{2^{(E-151)}}{2^{(E-127)}\cdot\left(1+\frac{M}{2^{23}}\right)}\cdot 100\%=\frac{2^{E}\cdot 2^{-151}}{2^{E}\cdot 2^{-127}\left(1+\frac{M}{2^{23}}\right)}\cdot 100\%=\frac{2^{-24}}{\left(1+\frac{M}{2^{23}}\right)}\cdot 100\%=$$

$$=\frac{1}{2^{24}+2M}\cdot 100\%=\frac{1}{16777216+2\cdot 5587427}\cdot 100\%=3{,}57755e-6\ (\%).$$

3) Представление третьего результата операции в формате IEEE754 – Single:
1 11110110 10101010100000111100011.
Экспонента числа равна 246 (больше нуля и меньше 255_{10} (11111111_2)) , следовательно, это число входит в диапазон нормализованных (отрицательных) чисел формата IEEE754 – Single.
Десятичное значение нормализованного числа (Single) вычисляется по формуле:

$$F=(-1)^{S}\cdot 2^{(E-127)}\cdot(1+M/2^{23})$$

$$F=(-1)^{1}\cdot 2^{(246-127)}\cdot(1+M/2^{23})=-2^{119}\cdot(1+5587427/8388608)\approx$$

$$\approx -1{,}666073\cdot 6{,}646140\cdot 10^{35}\approx -1{,}107295e+36$$

Шаг нормализованных чисел формата IEEE754 – Single: $h=2^{(E-150)}$. Максимальная абсолютная погрешность представления числа:
$\delta=h/2=2^{(E-151)}=2^{(246-151)}=2^{95}\approx 3{,}9614081e+28.$
Максимальная относительная ошибка (погрешность представления) нормализованного числа (Single):

$$\xi=\frac{1}{2^{24}+2M}\cdot 100\%=\frac{1}{16777216+2\cdot 5587427}\cdot 100\%=3{,}57755e-6\ (\%).$$

Ответ:

Число в формате Single	Десятичный эквивалент числа	Абсолютная погрешность представления	Относительная погрешность представления
0 00000000 10101010100000111100011	**7,822398e-39**	**7,006492e-46**	**8,94866e-6 %**
0 00000110 10101010100000111100011	**6,26707e-37**	**2,242077e-44**	**3,57755e-6 %**
1 11110110 10101010100000111100011	**-1,107295e+36**	**3,9614081e+28**	**3,57755e-6 %**

Далее должен быть приведен краткий анализ ответа с точки зрения величины (абсолютной) чисел и соответствующих им погрешностей.

7.2. Контрольные вопросы

1) Какие форматы используются для представления вещественных чисел в ЭВМ?
2) Какую характеристику типа с плавающей точкой определяет число разрядов порядка?
3) Какую характеристику типа с плавающей точкой определяет число разрядов порядка?
4) Что определяет знак порядка?
5) Что определяет знак мантиссы?
6) Какое представление вещественного числа считается нормализованным в математике?
7) В чем состоят особенности формата IEEE-754? Что представлено в каждом из трех полей (S, E, M) формата? Чем отличается представление Single от Double?
8) Что понимается под экспонентой числа, представленного в формате IEEE-754, и сколько разрядов она занимает (для Single и Double)?
9) Как вычислить истинный порядок числа, зная его экспоненту (для Single и Double)?
10) Что понимается под нормализованным числом в формате IEEE-754? Назовите границы диапазона нормализованных чисел.
11) Что понимается под остатком от мантиссы нормализованного числа, представленного в формате IEEE?
12) Сколько разрядов занимает остаток от мантиссы (для Single и Double)?
13) По какой формуле вычисляется десятичное значение нормализованного числа в формате IEEE-754?
14) Что понимается под денормализованным числом в формате IEEE-754? Для каких вещественных значений используется это представление? Назовите границы диапазона денормализованных чисел.
15) Чем представление денормализованных чисел в формате IEEE-754 отличается от представления нормализованных?
16) По какой формуле вычисляется десятичное значение денормализованного числа в формате IEEE-754?
17) Назовите все диапазоны чисел, которые представляет формат IEEE-754? Как определить в какой диапазон входит число по двоичному значению его экспоненты?
18) Как представляются в формате IEEE-754 ноль, плюс бесконечность и минус бесконечность.
19) Что в формате IEEE-754 понимается под NAN-значением?
20) Сформулируйте алгоритм сложения (вычитания) чисел с плавающей точкой.
21) Как вы думаете, почему порядки (экспоненты) слагаемых при выполнении операции сложения (вычитания) вещественных чисел выравниваются в сторону большего?
22) Сформулируйте правило вычитания экспонент (беззнаковых целых чисел) на сумматоре порядков.

23) Сформулируйте правило сложения мантисс на сумматоре мантисс (мантиссы представляются в дополнительных кодах с удвоенным знаковым разрядом).

24) Какие виды нарушения нормализации могут возникнуть при выполнении операций с плавающей точкой? Как осуществляется процедура нормализации для каждого вида?

25) Чем вызвано наличие погрешности в представлении вещественных чисел? К каким последствиям может привести «неаккуратный» алгоритм, использующий вычисления с плавающей точкой?

26) Как вычислить максимальную абсолютную и относительную ошибку представления вещественного числа, представленного в формате IEEE-754?

27) Как связана максимальная абсолютная ошибка представления вещественного числа в формате IEEE-754 с его экспонентой?

8. ПРАКТИЧЕСКАЯ РАБОТА 5. ОБРАБОТКА ДВОИЧНЫХ ВЕКТОРОВ. ПОРАЗРЯДНЫЕ ЛОГИЧЕСКИЕ ОПЕРАЦИИ. ОПЕРАЦИИ СДВИГОВ. ОПЕРАЦИИ СРАВНЕНИЯ И БУЛЕВОЙ ЛОГИКИ. ЗАДАНИЯ 14 – 18

8.1. Содержание, варианты и примеры выполнения заданий

Задание 14. Выполнить поразрядные логические операции И, ИЛИ, исключающее ИЛИ, НЕ над логическими векторами A и B. Варианты задания представлены в таблице 15.

Таблица 15 – Варианты для задания 14

НВ	A, B	НВ	A, B	НВ	A, B
1	10100101 11010011	11	11100011 10001110	21	00100101 11001011
2	11100111 10101011	12	01100111 11001001	22	01111001 10100010
3	11000011 10011001	13	00110111 01100100	23	10000111 10011001
4	10111101 01001001	14	01011110 10010010	24	11001001 00011010
5	10101010 00011001	15	11110101 10010010	25	11100110 01101001
6	01010101 11100010	16	01100110 11101000	26	10011101 01101001
7	11001100 01010110	17	10010011 00111010	27	11100001 10001001
8	11110000 01010011	18	10111110 11000100	28	10001000 10101110
9	00001111 10101100	19	10110110 01100101	29	11011100 01010110
10	11001110 01001001	20	01101010 11001100	30	10100011 00110110

Пример выполнения задания 14. A: 01010110; B: 10110010.

AND (&) $\wedge$

	0	1	0	1	0	1	1	0
	1	0	1	1	0	0	1	0
	0	0	0	1	0	0	1	0

XOR (^) $\oplus$

	0	1	0	1	0	1	1	0
	1	0	1	1	0	0	1	0
	1	1	1	0	0	1	0	0

OR (|) $\vee$

	0	1	0	1	0	1	1	0
	1	0	1	1	0	0	1	0
	1	1	1	1	0	1	1	0

NOT (~) A / $\overline{A}$

A	0	1	0	1	0	1	1	0
$\overline{A}$	1	0	1	0	1	0	0	1

NOT (~) B / $\overline{B}$

B	1	0	1	1	0	0	1	0
$\overline{B}$	0	1	0	0	1	1	0	1

Задание 15. Дан фрагмент java-программы. Представить значение результатов операций сдвигов C1, C2, C3, C4, C5, C6 (в двоичном виде).

…
int A = a; byte B = b; byte D = d;
int C1 = A >> D;
int C2 = A >>> D;
int C3 = A << D;
byte C4 = (byte) (B >> D);
byte C5 = (byte) (B >>> D);
byte C6 = (byte) (B << D);
… Варианты задания представлены в таблице 16.

Таблица 16 – Варианты для задания 15

НВ	a	b	d
1	10011111 00110011 11001100 11110011	11001100	00000011
2	11110000 11001100 10000001 11110111	11101100	00000100
3	10000001 00110011 01111110 10001111	10001111	00000101
4	11000111 01011100 10101010 10010011	11001101	00000010
5	11100101 11011011 01010001 10000111	11001000	00000001
6	10110100 00001111 00001111 11000001	10000011	00000110
7	10101100 11010101 00100100 00011001	10011101	00000011
8	11110011 00110011 00001111 01100011	10111001	00000100
9	10101111 00011100 00011100 10000111	11010101	00000101
10	11010011 10111011 00100100 00001111	10110011	00000010
11	10011111 00110011 11001100 11110011	11001100	00000100
12	11110000 11001100 10000001 11110111	11101100	00000101
13	10000001 00110011 01111110 10001111	10001111	00000010
14	11000111 01011100 10101010 10010011	11001101	00000001
15	11100101 11011011 01010001 10000111	11001000	00000110
16	10110100 00001111 00001111 11000001	10000011	00000011
17	10101100 11010101 00100100 00011001	10011101	00000100
18	11110011 00110011 00001111 01100011	10111001	00000101
19	10101111 00011100 00011100 10000111	11010101	00000010
20	11010011 10111011 00100100 00001111	10110011	00000001
21	10011111 00110011 11001100 11110011	11001100	00000101
22	11110000 11001100 10000001 11110111	11101100	00000010
23	10000001 00110011 01111110 10001111	10001111	00000001
24	11000111 01011100 10101010 10010011	11001101	00000110
25	11100101 11011011 01010001 10000111	11001000	00000011
26	10110100 00001111 00001111 11000001	10000011	00000100
27	10101100 11010101 00100100 00011001	10011101	00000101
28	11110011 00110011 00001111 01100011	10111001	00000010
29	10101111 00011100 00011100 10000111	11010101	00000100
30	11010011 10111011 00100100 00001111	10110011	00000101

Пример выполнения задания 15.
a: 10000111 01110011 11011100 11000011, b: 11101101, d: 00000100.

C1: 11111000 01110111 00111101 11001100
C2: 00001000 01110111 00111101 11001100
C3: 01110111 00111101 11001100 00110000

Результат любой арифметической и логической операции с byte- и short-операндами в Java имеет тип int.

При выполнении оператора **byte C4 = (byte) (B >> D);** сначала операнд B автоматически расширяется до int путем дублирования старшего разряда двоичного вектора (для чисел – это разряд знака) на 3 добавляемых байта. После этого производится сдвиг уже 32-битного вектора вправо на 4 разряда с доопределением освобождающихся слева разрядов значением старшего разряда. Затем производится «обрезка» полученного значения, в результате чего byte-переменной C4 присваивается младший байт результата операции сдвига.

Расширение B	**11111111 11111111 11111111 11101101**
B >> D	**11111111 11111111 11111111 11111110**
(byte) (B >> D)	**11111110**
C4:	**11111110**

Алгоритм выполнения оператора **byte C5 = (byte) (B >>> D);** аналогичен рассмотренному выше.

Расширение B	**11111111 11111111 11111111 11101101**
B >>> D	**00001111 11111111 11111111 11111110**
(byte) (B >>> D)	**11111110**
C5:	**11111110**

Показали, что на типах byte и short отличий в результатах операций « >> » и « >>> » не видно. Их можно увидеть только на более универсальном типе int.

Для операции **byte C6 = (byte) (B << D);** :

Расширение B	**11111111 11111111 11111111 11101101**
B << D	**11111111 11111111 11111110 11010000**
(byte) (B << D)	**11010000**
C6:	**11010000**

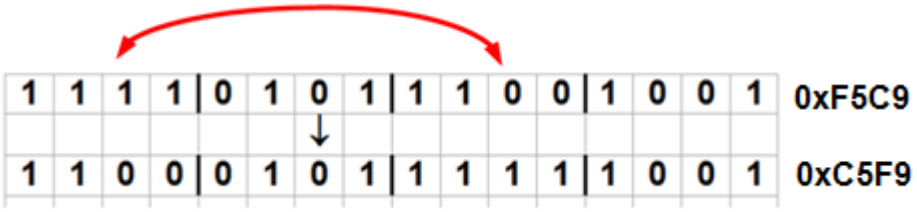
Задание 16. Разработать и промоделировать алгоритм выполнения заданной операции над двоичным вектором A. Представить соответствующий фрагмент java-программы и результаты его выполнения в окне кода BlueJ . Варианты задания представлены в таблице 17.

Таблица 17. – Варианты для задания 16

НВ	Количество байтов в двоичном векторе	Операция
1	1	Поменять местами тетрады
2	2	Поменять местами вторую и третью тетрады
3	2	Поменять местами первую и вторую тетрады
4	2	Поменять местами третью и четвертую тетрады
5	2	Поменять местами первую и третью тетрады
6	2	Поменять местами вторую и четвертую тетрады
7	3	Поменять местами первый и второй байт
8	3	Поменять местами второй и третий байт
9	3	Поменять местами первый и третий байт
10	3	Поменять местами первую и шестую тетрады
11	3	Поменять местами вторую и пятую тетрады
12	3	Поменять местами третью и четвертую тетрады
13	3	Поменять местами первую и третью тетрады
14	3	Поменять местами вторую и четвертую тетрады
15	3	Поменять местами третью и пятую тетрады
16	3	Поменять местами первую и четвертую тетраду
17	3	Поменять местами вторую и шестую тетрады
18	3	Поменять местами третью и шестую тетрады
19	4	Выполнить циклический сдвиг влево на 4 разряда
20	4	Выполнить циклический сдвиг вправо на 4 разряда
21	4	Выполнить циклический сдвиг влево на 8 разрядов
22	4	Выполнить циклический сдвиг вправо на 8 разрядов
23	4	Выполнить циклический сдвиг вправо на 16 разрядов
24	4	Выполнить циклический сдвиг влево на 16 разрядов
25	4	Поменять местами первый и третий байт
26	3	Выполнить циклический сдвиг влево на 4 разряда
27	2	Выполнить циклический сдвиг вправо на 4 разряда
28	4	Выполнить циклический сдвиг влево на 8 разрядов
29	1	Выполнить циклический сдвиг вправо на 3 разряда
30	3	Выполнить циклический сдвиг вправо на 5 разрядов

Пример выполнения задания 16.

Длина двоичного вектора − 16 битов. Поменять местами первую и третью тетрады вектора. Тестовый пример:



A: 1111 0101 1100 1001 (0xF5C9)

Алгоритм решения задачи (последовательность действий)

1) Выделить первую тетраду: 1111 0000 0000 0000
   B=A & 0xF000;
2) Сдвинуть вправо на 8 бит: **0000 0000 1111 0000**
   B=B >> 8;
3) Выделить третью тетраду: 0000 0000 1100 0000
   D=A & 0xF0;
4) Сдвинуть влево на 8 бит: **1100 0000 0000 0000**
   D=D << 8;
5) Объединить два полученных двоичных вектора:
   D=D | B; **1100 0000 1111 0000**
6) Обнулить первую и третью тетрады в A:
   A=A & 0xF0F; **0000 0101 0000 1001**
7) Объединить A и D:
   A=A | D **1100 0101 1111 1001**

При написании java-программы для хранения двоичных векторов выбираем тип int по следующим причинам:

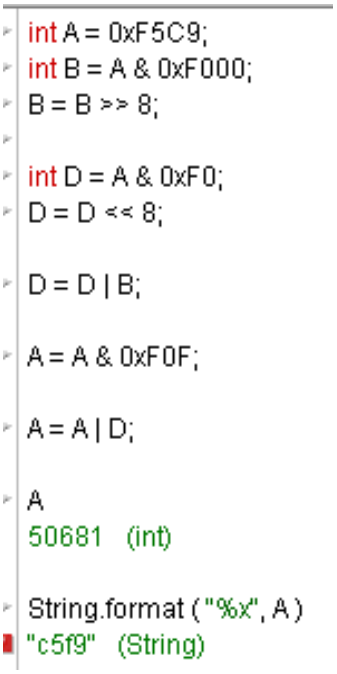
1) В short не поместится константа 0xF000 (не хватит одного бита, который отдан под знак). Максимальное значение для short – 0x7FFF (32767).
2) Операнды и результат при выполнении поразрядных логических операций все равно расширяются до int.

Два лишних старших нулевых байта в int-переменной просто не будут использоваться.

Фрагмент java-программы:

```
int A = 0xF5C9;
int B = A & 0xF000;
B = B >> 8;
int D = A & 0xF0;
D = D << 8;
D = D | B;
A = A & 0xF0F;
A = A | D;
```

Результаты выполнения в окне кода BlueJ:



Задание 17. В java-программе определены следующие переменные:

```
int a, b, c;
float d, e;
char ch1, ch2;
boolean f, f1, f2;
```

Определить значение переменной f при заданных значениях остальных переменных. Представить последовательность вычисления значения логического выражения, определяющего значение переменной f. Переписать выражение без лишних, по мнению студента, скобок. Обосновать. Что изменится, если в логическом выражении использовать удвоенные знаки логических операций (&& и ||)? Как значение переменной f будет представлено в компьютере?

Варианты для задания 17 приведены в таблице 18 (используются буквы только английского алфавита).

Таблица 18. – Варианты для задания 17

НВ	a	b	c	d	e	ch1	ch2	f1	f2	f
1	12	5	7	1,27	2,25	'F'	'0'	false	true	(a>(b+c)) | (d<e) & (ch1<ch2) | f1| (!(f1==f2))
2	6	8	10	5,25	3.25	'M'	'N'	true	false	(a<(b+c)) | (d>e) & (ch1<ch2) | !f1 | (f1!=f2) | ch1==88
3	10	29	17	21,5	3,2	'%'	'%'	true	true	((a>b) | (a>c)) & !(d>e) & (ch1=ch2) | f1 & f2
4	10	35	80	2,58	9,36	'a'	'b'	false	false	(a==b) | (a<c) & (d>e+3) | (ch1=ch2+1) & !(f1| f2)
5	11	15	14	98,1	15,3	'S'	'Y'	true	false	((a==b) | (a>c)) & (d<e*e) & ((ch1<ch2) | (f1==!f2))
6	4	9	1	6,89	3,54	'S'	'Y'	true	true	(c<b+a) | (d>b+a) & !(e<b) | (ch1<ch2) &(f1!= f2)
7	5	5	9	3,14	2,04	'1'	'+'	false	true	(c>21) | (a<b) & !(f1!=f2) & (ch1+2!=ch2) | (e<d+b)
8	7	98	16	56,1	46,5	'V'	'v'	false	false	(c-b>0) & (c<a) | (d-e>10,5) & !(ch1=='V' & ch2=='v') & (f1==f2)
9	10	16	30	2,75	7,98	'№'	':'	true	false	(c-5>b-10) & (d-2*e>0) | (a==0) & (ch2>ch1) | !(f1 | f2)
10	-5	-8	19	-2,5	-7,6	'Q'	'L'	true	true	((a>b+d) | (e>c-a)) & ((ch1<=ch2) | f1) & !f2
11	2	15	3	51,2	32,2	'F'	'0'	true	false	(a>(b+c)) | (d<e) & (ch1<ch2) | f1| (!(f1==f2))
12	54	-5	15	8,25	5.25	'N'	'M'	false	true	(a<(b+c)) | (d>e) & (ch1<ch2) | !f1 | (f1!=f2)
13	99	-5	19	9,51	8,14	'%'	'*'	false	false	((a>b) | (a>c)) & !(d>e) & (ch1=ch2) | f1 & f2
15	17	50	-3	-8,1	-5,3	'Y'	'S'	true	false	((a==b) | (a>c)) & (d<e*e) & ((ch1<ch2) | (f1==!f2))
16	14	99	-1	96,8	-3,5	'D'	'E'	false	false	(c<b+a) | (d>b+a) & !(e<b) | (ch1<ch2) &(f1!= f2)
17	31	15	45	-3,1	-9,3	'A'	'C'	false	true	(c>21) | (a<b) & !(f1!=f2) & (ch1+2!=ch2) | (e<d+b)
18	37	98	16	53,1	40,5	'D'	'v'	false	false	(c-b>0) & (c<a) | (d-e>10,5) & !(ch1=='V' & ch2=='v') & (f1==f2)
19	10	31	56	2,72	1,15	'@'	'#'	false	true	(c-5>b-10) & (d-2*e>0) | (a==0) & (ch2>ch1) | !(f1 | f2) | ch2<80

Продолжение таблицы 18

НВ	a	b	c	d	e	ch1	ch2	f1	f2	f
20	-5	-8	19	-2,5	-7,6	'Q'	'L'	false	false	((a>b+d) | (e>c-a)) & ((ch1<=ch2) | f1) & !f2
21	12	5	7	1,27	2,25	'F'	'0'	false	true	(a>(b+c)) | (d<e) & ((ch1<ch2) | f1| (!(f1==f2)))
22	6	8	10	5,25	3.25	'M'	'N'	true	false	((a<(b+c)) | (d>e)) & ((ch1<ch2) | !f1 | (f1!=f2))
23	10	29	17	21,5	3,2	'%'	'%'	true	true	(a>b) | (a>c) & !(d>e) & ((ch1=ch2) | f1) & f2
24	10	35	80	2,58	9,36	'a'	'b'	false	false	((a==b) | (a<c)) & ((d>e+3) | (ch1=ch2+1)) & !(f1| f2) | ch1>85
25	11	15	14	98,1	15,3	'S'	'Y'	true	false	(a==b) | (a>c) & (d<e*e) & (ch1<ch2) | (f1==!f2)
26	4	9	1	6,89	3,54	'S'	'Y'	true	true	((c<b+a) | (d>b+a)) & !(e<b) | (ch1<ch2) | (f1!= f2))
27	5	5	9	3,14	2,04	'1'	'+'	false	true	(c>21) & (a<b) | !(f1!=f2) & (ch1+2!=ch2) & (e<d+b)
28	7	98	16	56,1	46,5	'V'	'v'	false	false	(c-b>0) | (c<a) & ((d-e>10,5) | !(ch1=='V' | ch2=='v')) & (f1==f2)
29	10	16	30	2,75	7,98	'№'	':'	true	false	(c-5>b-10) | (d-2*e>0) & ((a==0) | (ch2>ch1) | !(f1 | f2))
30	-5	-8	19	-2,5	-7,6	'Q'	'L'	true	true	(a>b+d) & (e>c-a) | (ch1>=ch2) & f1 | !f2

Пример выполнения задания 17

a=15; b=20; c=30; d=3,75; e=9,98; ch1='N'; ch2='M';f1=true; f2=false;
f = (a < c+b) & (c > b) | (d – e > 0) & (ch2 == ch1+1) | !f1 & f2

При вычислении значения f нужно учитывать порядок действий в соответствующем выражении. Согласно приоритетам операции, используемые в Java, можно расположить следующим образом: 1) круглые и квадратные скобки; 2) операции инкремента (++), декремента (−−), побитовой инверсии (~) и логического отрицания (!), 3) арифметические операции умножения (*), деления (/), деления по модулю (%), 4) арифметические операции сложения (+) и вычитания (−), 5) операции сдвигов, 6) операции сравнения (>, >=, <,<=); 7) операции сравнения (==, !=); 8) побитовое или логическое И (&), 9) побитовое исключающее ИЛИ (^), 10) побитовое или

логическое ИЛИ (|), 11) короткое логическое И (&&), 12) короткое логическое ИЛИ (||), 13) троичная условная операция (?:), 14) операция присваивания (=).

Круглые скобки, имеющие наивысший приоритет позволяют изменить порядок действий в выражении.

$$\overset{2\ 1\ \ 9\ \ 3\ \ 12\ \ 4\ 5\ \ 10\ \ \ 7\ \ \ 6\ \ 13\ 8\ \ 11}{(a<c+b)\ \&\ (c>b)\ |\ (d-e>0)\&(ch2==ch1+1)\ |\ \ !\ f1\&\ f2}$$

1) $c+b:\quad 20+30=50;$

2) $a<c+b:\quad 15<50=true;$

3) $c>b:\quad 30>20=true;$

4) $d-e:\quad 3{,}75-9{,}98=-6{,}23;$

5) $d-e>0:\quad -6{,}23>0=\ false;$

6) $ch1+1:\quad 0\text{x}4D+1=0\text{x}4E;$

7) $ch2==ch1:\quad 0\text{x}4E==0\text{x}4E=true;$

8) $!f1:\quad !true=\ false;$

9) $\underbrace{a<c+b}_{true}\ \&\ \underbrace{c>b}_{true}:\quad true\ \&\ true=true;$

10) $\underbrace{d-e>0}_{false}\ \&\ \underbrace{ch2==ch1+1}_{true}:\quad false\ \&\ true=\ false;$

11) $\underbrace{!f1}_{false}\ \&\ f2:\quad false\ \&\ false=\ false;$

12) $\underbrace{d-e>0}_{false}\ \&\ \underbrace{ch2==ch1+1}_{true}:\quad false\ \&\ true=\ false;$

13) $\underbrace{a<c+b\ \&\ c>b}_{true}\ |\ \underbrace{d-e>0\ \&\ ch2==ch1+1}_{false}:\quad true\ |\ false=\ false;$

14) $\underbrace{a<c+b\ \&\ c>b\ |\ d-e>0\ \&\ ch2==ch1+1}_{false}\ |\ \underbrace{!f1\&\ f2}_{false}:\quad false\ |\ false=\ false;$

Анализируя приоритеты операций, можно сделать вывод, что в рассматриваемом выражении можно опустить все скобки. Результат при этом не изменится, хотя последовательность выполнения тех же действий изменится: первым станет восьмое, вторым – первое, третьим – четвертое, четвертым – шестое, пятым – второе, шестым – третье, седьмым – пятое, восьмым – седьмое, последовательность остальных действий не изменится:

$$\overset{5\ \ 2\ \ 9\ \ 6\ \ 12\ \ 3\ \ 7\ \ 10\ \ \ \ 8\ \ \ \ 4\ \ 13\ 1\ \ \ 11}{a<c+b\&c>b\ |\ d-e>0\&ch2==ch1+1\ |\ \ !\ f1\&\ f2}$$

Использование коротких логических операций позволяет сделать их выполнение более быстрым: если значение левого операнда полностью определяет значение, возвращаемое операцией (для && − false, для || − true), то правый операнд не вычисляется.

Если в исходном выражении использовать короткие логические операции (&& и ||), то его можно записать так:

$$\overset{2}{(}a\overset{1}{<}c\overset{9}{+}b\overset{3}{)}\overset{12}{\&\&}\overset{4}{(}c\overset{5}{>}b\overset{10}{)}\overset{7}{\|}\overset{6}{(}d\overset{13}{-}e\overset{8}{>}0\overset{11}{)}\&\&(ch2==ch1+1)\|\ !f1\&\&f2$$

Тогда правый операнд не будет вычисляться в десятом, одиннадцатом, двенадцатом и тринадцатом действиях.

В памяти компьютера переменная типа boolean занимает 1 байт. Если значение переменной равно false, то все биты этого байта равны 0. Любая другая комбинация значений битов интерпретируется как true. Переменная f после присвоения ей значения выражения будет представлена в памяти компьютера следующим образом:

f	0	0	0	0	0	0	0	0

Задание 18. Указать, какие из операторов вызовут ошибки (связанные с некорректным использованием типов данных) в java-программе, а какие нет? Обосновать. Варианты для задания 18 приведены в таблицах 19, 20.

Таблица 19 – Варианты для задания 18

НВ	Номера операторов (последовательностей операторов) из таблицы 20
1	1, 2, 11, 12, 21, 22, 23, 26, 35, 43, 44
2	3, 4, 13, 14, 27, 28, 37, 40, 41, 44, 47
3	5, 6, 15, 16, 29, 30, 45, 46, 49, 51, 54
4	8, 9, 17, 18, 23, 24, 25, 29, 32, 41,48
5	10, 11, 19, 20, 31, 34, 37, 38, 39, 45, 50
6	3, 5, 9, 10, 18, 21, 30, 41, 42, 51, 53
7	4, 7, 12, 17, 20, 23, 32, 33, 35, 26, 52
8	1, 7, 8, 16, 19, 28, 31, 36, 37, 43, 40
9	10, 11, 18, 29, 30, 35, 44, 45, 47, 51, 54
10	13, 14, 19, 20, 23, 25, 31, 32, 41, 46, 49
11	2, 8, 9, 13, 14, 24, 27, 35, 36, 37, 39
12	1, 4, 9, 10, 15, 29, 34, 38, 50, 51, 53
13	8, 11, 14, 23, 26, 27, 30, 33, 35, 36, 42
14	10, 14, 15, 27, 31, 32, 40, 45, 48, 52
15	1, 6, 9, 16, 27, 32, 43, 45, 50, 51, 54,
16	3, 10, 15, 18, 21, 23, 25, 32, 41, 48, 47
17	11, 13, 14, 27, 30, 33, 36, 37, 39, 49, 50
18	5, 6, 14, 19, 24, 31, 34, 41, 44, 51, 53
19	7, 8, 11, 16, 19, 23, 26, 36, 38, 45, 46
20	3, 4, 9, 16, 17, 32, 37, 40, 41, 42, 48

Продолжение таблицы 19

НВ	Номера операторов (последовательностей операторов) из таблицы 19
21	5, 10, 13, 18, 29, 33, 34, 50, 51, 52, 54
22	2, 7, 10, 20, 21, 23, 25, 29, 30, 35, 43
23	6, 9, 15, 22, 31, 32, 33, 37, 39, 46, 47
24	5, 10, 11, 14, 21, 28, 36, 45, 49, 51, 53
25	8, 11, 17, 22, 23, 24, 26, 29, 33, 36, 50,
26	9, 16, 19, 29, 30, 37, 38, 40, 41, 44, 48
27	4, 7, 16, 17, 27, 30, 35, 42, 50, 51, 54
28	5, 6, 15, 23, 28, 29, 35, 36, 48, 25, 52
29	2, 11, 14, 21, 28, 31, 37, 39, 41, 46, 47
30	5, 10, 13, 22, 33, 44, 45, 49, 50, 51, 53

Таблица 20 – Данные для вариантов задания 18

№	Операторы	№	Операторы	№	Операторы
1	int n = 0937;	5	double a = 5; float b =a;	9	short n = -33768;
2	float a = 2.5f;	6	float a = 2.5f;	10	boolean f = 'A' > 'B';
3	int m = 0x9ag;	7	float a = 3.5f; long i = a;	11	short n = 32768;
4	double b = 2.5;	8	double b = 23.568E15;	12	boolean f = 23 < 15;
13	int n = 0x7FFFFFFFFFFFFFFFL;	27	byte b = 0x3F; b = b >> 3;	41	byte c= 300;

14	char ch1 = 'A';
	char ch2 = 'B';
	char ch3 = (char)
	(ch1+ch2);
		28	float a = (float) 2.5;	42	boolean f1 = true;
					boolean f2 = false;
					boolean f3 = f1==f2;

15	long n =
	0x7FFFFFFFFFFFFFFF;
		29	short a = 3678;
			a = a / 3;
				43	boolean f1 = true;
					boolean f2 = false;
					boolean f3 = f1<f2;

16	int a = 127; byte b = a;	30	double b = 927.325E-3;	44	double b = 3.5D;

17	int n = 7FFFFFFFFF;	31	char ch1 = 'A';
			char ch2 = 'B';
			char ch3 = ch1+ch2;
				45	int a = 1, b = 2;
					double c = 2.5;
					double d = a / b + c;

18	int a = 57365; long b = a;	32	double b = 1024.75d;	46	float a = 25,349F

19	long n =
	0x7fffffffffffffffffL;
		33	int b = 4147483647;	47	double a = 3.5;
					double b = 2.5;
					double c = 1/(a+b);

Продолжение таблицы 20

№	Операторы	№	Операторы	№	Операторы
20	int n = 0354;	34	double a = 127.351; float b = (float) a;	48	float a = 357.52; long n = (float) a;

21	float a = 2.5;	35	int s = -2500483648;	49	double a = 123.5;
					double b = -912.5;
					double c = 1/2*(a+b);

22	short m = -32768;	36	byte b = 0x3f; b = (byte) (b << 4);	50	short a = 3678; a = (short) a / 3;

23	double a = 2.0,
	b = 3.0, h = 0.1;
	double x, f = 0.0;
	for (x=a; x!=b; x=x+h)
	{y =x*x-7; …};
		37	double a = 2.0,
			b = 3.0, h = 0.1;
			double f = 0.0;
			double x = a;
			do { f = x * x ; …
			x = x + h; }
			while (x != b);
				51	double a = 2.0,
					b = 3.0, h = 0.1;
					double f = 0.0;
					double x = a;
					while (x != b)
					{ f = x * x ; …
					x = x + h; }

24	double a = 1, b = 2;
	double c = 2.5;
	double d = a / b + c;
		38	double a = 3.5;
			double b = 2.5;
			double c = 1.0/(a+b);
				52	double a = 123.5;
					double b = -912.5;
					double c =
					1.0/2*(a+b);

25	int a=1, b=3, c=6; if (a+b+c)…	39	double a =1.2, b=3.5; if ((a+b) & (a-b))…	53	char d = 80, e = 'A'; if ((d+1) | (e-1))…

26	int a=1, b=3, c=6;
	if ((a>b) & (b<c))…
		40	double a =9.7, b=-3.6;
			if ((a+b) > 0 &
			(a-b) < 0)…
				54	char d = 88, e = 'V';
					boolean f = true;
					if ((d<e) | f)…

Пример выполнения задания 18.

Номера операторов (последовательностей операторов) из таблицы 19:

7, 14, 15, 27, 31, 32, 40, 42, 43, 45, 48, 51, 53, 54 *(взяты для примера, не соответствуют реальным номерам).*

№	Исходная последовательность операторов	Анализ исходной последовательности операторов	Исправленная последовательность операторов
7	float b = 674.25f; long m = b;	Данная последовательность операторов повлечет за собой ошибку компиляции, т.к. не определено автоматическое преобразование из float в long. Требуется явное преобразование типа	float b = 674.25; long m = (long) b;

14	byte m = 45; int n = m;	Последовательность не вызовет ошибки. Будет выполнено автоматическое расширяющее преобразование из byte в int.	–
15	int s = −64; byte q = s;	Последовательность не вызовет ошибки, т.к. значение int-переменной *s* (−64) входит в диапазон типа byte, в противном случае компилятор выдал бы ошибку.	–
27	long k = 0x7FFFFFFFFFFFFFF;	Последовательность вызовет ошибку компиляции, т.к. тип используемого литерала – int, а значение литерала не помещается в 32 битах. Для исправления ошибки нужно явно указать, что тип литерала – long (добавить *l* или *L* в конце записи литерала).	long k = 0x7FFFFFFFFFFFFFFL;
31	float h = (float) 22.7;	Оператор не вызовет ошибки, т.к. указано явное преобразование во float литерала типа double.	–
32	byte b = 0x5F; b = b << 4;	Второй оператор последовательности вызовет ошибку компиляции, т.к. результат операции сдвига, имеющий тип int, присваивают переменной типа byte, которая на 3 байта короче int.	byte b = 0x5F; b = (byte)(b << 4);

40	boolean a = false;
	boolean b = true;
	boolean c = a > b;
		Последовательность операторов вызовет ошибку, т.к. из всех операций сравнения для boolean определены только операции равенства и неравенства.	boolean a = false;
			boolean b = false;
			boolean c = a != b;

42	int n = 01678;	Оператор вызовет ошибку компиляции, т.к. с ведущего нуля начинается восьмеричный литерал, и цифры 8 в его записи быть не может.	int n = 01677

43	double b = 927.325E-3;	Оператор не вызовет ошибки. Вещественный литерал, представленный в научном формате, имеет тип double и присваивается переменной того же типа.	–
45	int r = −2247583647;	Оператор вызовет ошибку компиляции, т.к. значение литерала, имеющего тип int выходит за границы диапазона int (от −2147483648 до 2147483647)	long r = −2247583647;
48	float a = 123.56;	Оператор вызовет ошибку компиляции «Возможна потеря точности», т.к. значение вещественного литерала, имеющего по умолчанию тип double (64 бита), присваивается переменной типа float (32 бита). Нужно явно указать тип литерала или сделать явное преобразование типа.	float a = 123.56f; или float a = 123.56F; или float a = (float)123.56;

51	double a = -1.5, b = 1.5, h = 0.2;
	double x, f = 0.0;
	for (x=a; x!=b; x=x+h)
	{y =x*x+3x-10; …};
		Данная последовательность операторов вызовет ситуацию зацикливания во время выполнения программы. Это связано с тем, что операции над вещественными числами (x=x+h) выполняются с погрешностью, вызванной ограниченной точностью представления чисел в ЭВМ. Условие выхода из цикла (x!=b) никогда не выполнится.	double a = -1.5, b = 1.5, h = 0.2;
			double x, f = 0.0;
			for (x=a; x < b; x=x+h)
			{y =x*x+3x-10; …};

53	double a = 45.38;
	double b = −0.5;
	double c = 5/24*(a*a+b*b+2.45);
		Программа выполнится успешно, но результат будет неверный (равен нулю), т.к. 5 на 24 делится по правилам целочисленного деления (частное 0 , остаток 5).	double a = 45.38;
			double b = −0.5;
			double c = 5.0/24.0 *(a*a+b*b+2.45);

54	int a=234, b=135; if (a < b−10)…	Ошибок нет, после if в скобках записано логическое выражение, которое может принимать значения true или false (при данных значениях a и b – false), что соответствует java-синтаксису оператора if.	–

8.2. Контрольные вопросы

1) Что такое двоичный вектор?
2) Какие типы данных используются для хранения двоичных векторов в Java?
3) Какие поразрядные логические операции определены для этих типов? Назовите правила их выполнения.
4) Какие операции сдвига определены в Java? Сколько операндов у операции сдвига и какие? В чем особенности выполнения логического сдвига вправо?
5) Почему для хранения двоичных векторов, длина которых меньше или равна 32 битам удобно использовать тип int?
6) Назовите элементарные задачи анализа и обработки двоичных векторов? Как они решаются?
7) Сформулируйте алгоритм выполнения перестановки групп битов.
8) Сформулируйте алгоритм выполнения циклического сдвига n-разрядного вектора на m разрядов влево (вправо), при условии m < n.
9) Для чего используется тип char. Сколько битов занимает значение типа char?
10) Что такое Unicode? Как представляется символ в Java? Сколько символов можно представить соответствующим образом?
11) Как можно задать символьный литерал в программе? Назовите 3 способа.
12) По какому принципу выполняется сравнение символов? Чему равны значения выражений 'A'<'D' , 'C'>'F' , '9'<'A'. Какого они типа?
13) Для чего используется тип boolean? Какой класс оболочка ему соответствует в Java?
14) Какие значения может принимать тип boolean?
15) Чему равны значения выражений 2>5 , 3<10 ?
16) Какие операции определены для данных логического типа (boolean)?
17) Назовите правила выполнения операций булевой логики.
18) В чем состоит особенность выполнения укороченных операций && и || ?
19) Что из себя представляет условное выражение?
20) Приведите пример сложного условного выражения. Чему равно его значение?
21) Какие ошибки в программах могут быть вызваны некорректным использованием типов данных? Как вы думаете, к каким типам относятся эти ошибки (ошибки, обнаруживаемые компилятором, ошибки времени выполнения, логические)?

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Грошев А. С. Информатика : учебник для вузов / А. С. Грошев. – Архангельск : Арханг. гос. техн. ун-т, 2010. – 470 с. — Текст : непосредственный.
2. Бойченко Л. П. Арифметические и логические основы компьютеров и дискретных автоматов: учеб. пособие/ Л. П. Бойченко, З. Х. Ягубов, Н. М. Выборова, Т. А. Туманова. – Ухта : УГТУ, 2011. – 100 с. — Текст : непосредственный.
3. Апраксин Ю. К. Основы теории проектирования цифровых автоматов : учеб. пособие для вузов/ Ю. К. Апраксин. – Севастополь : СевГТУ, 2001. – 345 с. — Текст : непосредственный.
4. Ноутон П. Java™ 2 / П. Ноутон, Г. Шилдт. – СПб. : БХВ-Петербург, 2008. – 1072 с. — Текст : непосредственный.
5. Яшкардин В. Л. IEEE 754 − стандарт двоичной арифметики с плавающей точкой. – SoftElectro. Промышленная электроника и программирование. URL : http://www.softelectro.ru/ieee754.html - Текст : электронный.
6. Таблица символов Юникода® – URL : https://unicode-table.com/ru/#control-character. — Текст : электронный.

ПРИЛОЖЕНИЕ А
Пример оформления титульного листа отчета

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования «Севастопольский государственный университет»

Институт информационных технологий
Кафедра информационных технологий и компьютерных систем

ОТЧЕТ

о выполнении цикла практических работ

«АРИФМЕТИЧЕСКИЕ ОСНОВЫ КОМПЬЮТЕРОВ. ПРЕДСТАВЛЕНИЕ И ОБРАБОТКА ДАННЫХ ПРОСТЫХ ТИПОВ В ЭВМ»

по дисциплине «Алгоритмизация и программирование вычислительных процессов»

Вариант 15

Выполнил
студент группы ИВТ/б-11-о
Орлов И.В.
Проверил
доцент Волкова Т.В.

Севастополь
202_

АРИФМЕТИЧЕСКИЕ ОСНОВЫ КОМПЬЮТЕРОВ. ПРЕДСТАВЛЕНИЕ И ОБРАБОТКА ДАННЫХ ПРОСТЫХ ТИПОВ В ЭВМ

Методические указания
к выполнению цикла практических работ
по дисциплине «Алгоритмизация

Составители: **Волкова** Татьяна Викторовна,
Владимирова Елена Сергеевна

В авторской редакции

Изд. № 68/2023. Объем 3,75 п.л. Усл. печ. л. 3,49. Уч.-изд. л. 3,675.
РИИЦМ ФГАОУ ВО «Севастопольский государственный университет»