ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

УПРАВЛЕНИЕ ДИСТАНЦИОННОГО ОБУЧЕНИЯ И ПОВЫШЕНИЯ КВАЛИФИКАЦИИ

Кафедра «Вычислительные системы и информационная безопасность»

Методические указания

к проведению лабораторной работы №1
по дисциплине
«Аппаратные средства вычислительной техники»

«Представление данных в ЭВМ и машинная арифметика»

Автор
Айдинян А.Р.

Ростов-на-Дону, 2018

Аннотация

«Методические указания» предназначены для студентов очной формы обучения направления подготовки 10.03.01 «Информационная безопасность» для выполнения лабораторной работы №1 по дисциплине «Аппаратные средства вычислительной техники»

Автор

доцент, к.т.н.,
доцент кафедры «Вычислительные системы и информационная безопасность»
Айдинян А.Р.

Оглавление

Представление данных в ЭВМ и машинная арифметика...4

Краткие теоретические сведения..... 4
Порядок выполнения работы.....15
Варианты индивидуальных заданий 15
Контрольные вопросы 16
Список литературы.....17

ПРЕДСТАВЛЕНИЕ ДАННЫХ В ЭВМ И МАШИННАЯ АРИФМЕТИКА

Цель работы: Изучить способы кодирования целых и вещественных чисел и способы выполнения арифметических операций в двоичном коде.

Краткие теоретические сведения

Двоичные коды.

Для представления информации в памяти ЭВМ (как числовой, так и нечисловой) используется двоичный способ кодирования.

Для записи двоичного числа на бумаге кроме символов '0' и '1' применяются символы '+', '–', двоичная запятая. Однако при двоичном кодировании (представлении числа в памяти компьютера) используются только цифры 0 и 1.

В компьютере для хранения чисел и других значений используются ячейки ОЗУ. Элементарная ячейка ОЗУ имеет длину 8 битов (1 байт). Наибольшую последовательность битов, которую ЭВМ может обрабатывать как единое целое, называют *машинным словом*. Длина машинного слова зависит от разрядности процессора и может быть равной 16, 32, 64 битам и т.д.

Разряды (биты слова) нумеруются справа налево, начиная с 0. На рис. 1 показана нумерация битов в двухбайтовом машинном слове.

№ бита	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Двоичный код	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1

Рис. 1. Нумерация битов в двухбайтовом машинном слове

В двоичном коде набор символов ограничен нулем и единицей и занимает определенное количество разрядов, поэтому:

- для записи положительных и отрицательных чисел не используются символы '+' и '–';
- двоичная запятая не ставится, а подразумевается между определенными разрядами двоичного кода;
- незначащие разряды двоичного числа в двоичном коде заполняются нулями, либо повторением знака числа в зависимости от вида двоичного кода.

Виды двоичных кодов для целых чисел.

Целые беззнаковые коды. Диапазон значений величин зависит от количества битов памяти, отведенных для их хранения. Для беззнакового представления целых двоичных числа изменяются от 0 до $2^n - 1$, а в представлении со знаком — от $\left(-2^{n-1}\right)$ до

$\left(2^{n-1}-1\right)$.

Для определённости примем длину слова процессора равной восьми битам. В этих кодах каждый двоичный разряд представляет собой степень цифры 2:

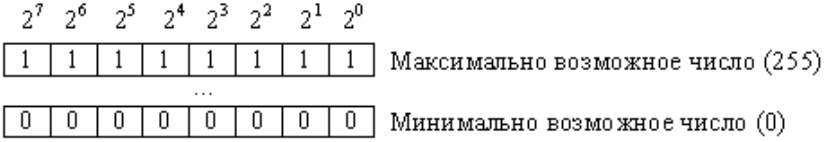

Рис. 2. Целые беззнаковые коды

На рис. 2 над каждым разрядом беззнакового двоичного кода приведено значение его веса. При этом минимально возможное число, которое можно записать таким двоичным кодом, равно 0. Максимально возможное число, которое можно представить этим кодом, определяется следующей формулой:

$M=2^n-1$,

где n – разрядность двоичного числа. Разрядность числа обычно выбирают кратной разрядности микропроцессора.

В случае двоичного 8-разрядного беззнакового двоичного кода целые числа, которые можно записать с его помощью, находятся в диапазоне от 0 до 255. Восьмиразрядное двоичное число обычно называют байтом иногда его еще называют октетом или полусловом.

Для беззнакового двоичного 16-разрядного кода диапазон представляемых значений будет от 0 до 65535. В микропроцессорной системе, построенной на 8-разрядном процессоре, для хранения 16-разрядного числа используются две ячейки памяти, расположенные в соседних адресах.

В качестве примера записи 16-разрядного числа в двух соседних 8 разрядных ячейках памяти, на рис. 3 приведена запись числа $56249_{(10)} = 1101101110111001_{(2)}$.

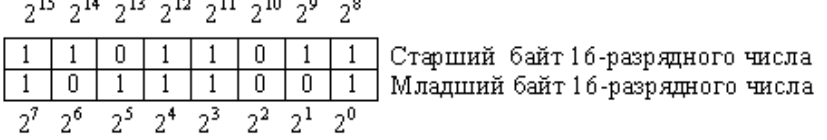

Рис. 3. Формат 16-разрядного беззнакового двоичного кода, записанного в двух соседних ячейках памяти

Следует отметить, что в отличие от действий в математике, при записи числа в прямом двоичном коде не существует возможности не записывать старшие незначащие нули. Поэтому число $12_{(10)}$ не может быть записано как $1100_{(2)}$. Оно обязательно долж-

но быть записано во всех восьми разрядах кода: 00001100. Если число первоначально было определено как 16-разрядное, то в этом случае это же самое число $12_{(10)}$ должно быть записано как $0000000000001100_{(2)}$. В качестве еще одного примера рассмотрим запись числа $31_{(10)}$. Оно будет записано в 8-разрядном прямом беззнаковом двоичном коде как $00011111_{(2)}$, а в 16 разрядном – 0000000000011111.

Прямые целые знаковые двоичные коды. В этих кодах старший разряд в слове используется для представления знака числа.

В прямом знаковом коде нулем обозначается знак '+', а единицей – знак '-'. В случае представления величины со знаком самый левый (старший) разряд указывает на положительное число, если содержит нуль, и на отрицательное, если – единицу.

В результате введения знакового разряда диапазон чисел смещается в сторону отрицательных чисел. Формат 8-разрядного прямого знакового двоичного кода приведен на рис. 4. На рис. приведено шесть различных чисел, записанных в этом коде. Слева указаны десятичные числа, соответствующие двоичному представлению обратного кода, а сверху, над каждым разрядом двоичного кода, указан его вес.

Рис. 4. Прямые целые знаковые коды

В случае двоичного восьмиразрядного знакового целого числа диапазон чисел, которые можно записать таким кодом: -127 .. +127. Для шестнадцатиразрядного кода этот диапазон будет: -32767 .. +32767. В восьмиразрядном процессоре для хранения такого числа тоже используется две ячейки памяти, расположенные в соседних адресах.

Обратите внимание, что при считывании содержимого памяти микропроцессора мы ни по каким внешним признакам не

сможем отличить знаковый двоичный код от беззнакового. Это сможет сделать только программа, в которой заложена информация о том, в каких ячейках памяти какой код следует использовать для хранения чисел. В результате человек или программа, не обладающие подобной информацией, при попытке прочитать ячейки памяти не смогут узнать – какое же число записано в данных конкретных ячейках памяти.

Недостатком такого кода является то, что знаковый разряд и цифровые разряды приходится обрабатывать раздельно. Алгоритм программ, работающий с такими кодами получается сложный. Для выделения и изменения знакового разряда приходится применять механизм маскирования разрядов, что резко увеличивает размер программы и уменьшает ее быстродействие. Для того, чтобы алгоритм обработки знакового и цифровых разрядов не различался, были введены обратные двоичные коды.

Обратные двоичные коды. Обратные двоичные коды отличаются от прямых только тем, что отрицательные числа в них получаются инвертированием всех разрядов числа. При этом знаковый и цифровые разряды не различаются. Алгоритм работы с такими кодами резко упрощается.

Формат 8-разрядного обратного знакового двоичного кода приведен на рис. 5. На рис. 5 приведено шесть различных чисел, записанных в этом коде. Слева указаны десятичные числа, соответствующие двоичному представлению обратного кода.

Рис. 5. Целые обратные двоичные коды

Обратите внимание, что знак числа при инвертировании получается автоматически. При инвертировании положительного числа, в котором знак '+' обозначен логическим нулем, мы получаем логическую единицу, то есть знак '-'.

В этом коде, точно также, как и в прямом знаковом двоич-

ном коде можно записывать как 8-разрядные, так и шестнадцати или 32-разрядные двоичные числа. При этом потребуется использовать несколько ячеек оперативного запоминающего устройства, как это иллюстрировалось на рис. 6 для прямого беззнакового двоичного кода.

Тем не менее, при работе с обратными кодами требуется специальный алгоритм распознавания знака, вычисления абсолютного значения числа, восстановления знака результата числа. Кроме того, в прямом и обратном коде числа для запоминания числа 0 используется два кода тогда, как известно, что число 0 положительное и отрицательным не может быть никогда.

Дополнительные двоичные коды. От перечисленных недостатков свободны дополнительные коды. Эти коды позволяют непосредственно суммировать положительные и отрицательные числа, не анализируя знаковый разряд и при этом получать правильный результат. Все это становится возможным благодаря тому, что дополнительные числа являются естественным кольцом чисел, а не искусственным образованием как прямые и обратные коды. Кроме того, немаловажным является то, что вычислять дополнение в двоичном коде чрезвычайно легко. Для этого достаточно к обратному коду добавить 1. То есть для получения отрицательного числа в дополнительном коде необходимо модуль числа проинвертировать и прибавить 1.

Рис. 6. Дополнительные двоичные коды

Диапазон чисел, которые можно записать таким кодом: -128 .. +127. Для шестнадцатиразрядного кода этот диапазон будет: -32768 .. +32767. В восьмиразрядном процессоре для хранения такого числа используется две ячейки памяти, расположенные в соседних адресах.

В памяти объемом n битов может храниться 2^n различных

значений. Например, величины типа Integer лежат в диапазоне от –32768 (-2^{15}) до 32767 ($2^{15}-1$) и для их хранения отводится 2 байта (16 битов); типа Long — в диапазоне от -2^{31} до 2^{31-1} и размещаются в 4 байтах (32 битах).

Кодирование символов.

Набор символов персональных ЭВМ, совместимых с IBM PC, чаще всего является расширением кода ASCII. В этом случае для кодирования символов достаточно одного байта, что позволяет представить 256 символов (с десятичными кодами от 0 до 255). В настоящее время также используются и двухбайтовые представления символов.

Представление целых чисел в дополнительном коде. Дополнительный код положительного числа совпадает с его прямым кодом. Прямой код целого числа может быть получен следующим образом: число переводится в двоичную систему счисления, а затем его двоичную запись слева дополняют таким количеством незначащих нулей, сколько требует тип данных, к которому принадлежит число. Например, если число $28_{(10)} = 11100_{(2)}$ объявлено величиной типа Integer (шестнадцатибитовое со знаком), то его прямым кодом будет 0000000000011100, а если величиной типа Long (тридцатидвухбитовое со знаком), то его прямой код будет 00000000000000000000000000011100. Для более компактной записи чаще используют шестнадцатеричное представление кода. Полученные коды можно переписать соответственно как $001C_{(16)}$ и $0000001C_{(16)}$.

Дополнительный код целого отрицательного числа может быть получен следующим образом:

1) записать прямой код модуля числа;

2) инвертировать его (заменить единицы нулями, нули — единицами);

3) прибавить к инверсному коду единицу.

Например, запишем дополнительный код числа (–28), интерпретируя его как величину типа Long (тридцатидвухбитовое со знаком):

1) прямой код числа $28_{(10)}$ есть 00000000000000000000000000011100;

2) инверсный код — 11111111111111111111111111100011;

3) дополнительный код — 11111111111111111111111111100100 или $FFFFFFE4_{(16)}$.

При получении числа по его дополнительному коду, прежде всего, необходимо определить его знак. Если число окажется положительным, то просто перевести его код в десятичную систему

счисления. В случае отрицательного числа необходимо выполнить следующие действия:

1) вычесть из кода числа 1;

2) инвертировать код;

3) перевести в десятичную систему счисления. Полученное число записать со знаком минус.

Примеры. Запишем числа, соответствующие дополнительным кодам.

Дано число $0000000000011111_{(2)}$. Поскольку в старшем разряде записан нуль, то результат будет положительным. Это код числа $31_{(10)}$.

Дано число $1111111110000001_{(2)}$. Здесь записан код отрицательного числа. Вычитаем из кода числа 1:

$1111111110000001_{(2)} - 1_{(2)} = 1111111110000000_{(2)}$.

Инвертируем код: 0000000001111111. Далее переводим инвертированный код в десятичную систему счисления $1111111_{(2)} = 127_{(10)}$. Ответ: (–127).

В обратных и дополнительных кодах наблюдается интересный эффект, который называется эффект распространения знака. Он заключается в том, что при преобразовании однобайтного числа в двухбайтное достаточно всем битам старшего байта присвоить значение знакового бита (старшего бита) младшего байта. То есть для хранения знака числа можно использовать сколько угодно старших бит. При этом значение кода совершенно не изменяется.

Контроль переполнения при выполнении арифметических операций.

Использование для представления знака числа двух бит предоставляет возможность контролировать переполнения при выполнении арифметических операций. Рассмотрим несколько примеров. В них число кодируется 8 битами и добавляется дополнительный девятый бит переноса, который должен совпадать с битом знака (восьмым битом).

1) Просуммируем числа 12 и 5

$$\begin{array}{r} 000001100 \\ +\ 000000101 \\ \hline 000010001 \end{array} \Leftrightarrow \begin{array}{r} +12 \\ +\ 5 \\ \hline 17 \end{array}$$

Перенос Знак

В этом примере видно, что в результате суммирования получается правильный результат. Это можно проконтролировать

по флагу переноса C, который совпадает со знаком результата (действует эффект распространения знака).

2) Просуммируем два отрицательных числа -12 и -5

$$\begin{array}{r} 11111010\,0 \\ +\,11111101\,1 \\ \hline 1\,11101111 \end{array} \Leftrightarrow \begin{array}{r} -12 \\ +\,-5 \\ \hline -17 \end{array}$$

В этом примере флаг переноса C тоже совпадает со знаком результата, то есть переполнения не произошло и в этом случае

3) Просуммируем положительное и отрицательное число -12 и +5

$$\begin{array}{r} 11111010\,0 \\ +\,00000010\,1 \\ \hline 11111100\,1 \end{array} \Leftrightarrow \begin{array}{r} -12 \\ +\,+5 \\ \hline -7 \end{array}$$

В этом примере при суммировании положительного и отрицательного числа автоматически получается правильный знак результата. В данном случае знак результата отрицательный. Флаг переноса совпадает со знаком результата, поэтому переполнения не было (мы можем убедиться в этом непосредственными вычислениями на бумаге или на калькуляторе).

4) Просуммируем положительное и отрицательное число +12 и -5

$$\begin{array}{r} 00000110\,0 \\ +\,11111101\,1 \\ \hline 00000011\,1 \end{array} \Leftrightarrow \begin{array}{r} +12 \\ +\,-5 \\ \hline +7 \end{array}$$

В данном примере знак результата положительный. Флаг переноса совпадает со знаком результата, поэтому переполнения не было и в этом случае.

5) Просуммируем числа 100 и 31

$$\begin{array}{r} 00110010\,0 \\ +\,00001111\,1 \\ \hline 01000001\,1 \end{array} \Leftrightarrow \begin{array}{r} +100 \\ +\,+31 \\ \hline +131!! \end{array}$$

В этом примере видно, что в результате суммирования произошло переполнение восьмибитовой переменной, т.к. в результате операции над положительными числами получился отрицательный результат. Однако если рассмотреть флаг переноса, то он не совпадает со знаком результата. Эта ситуации является признаком переполнения результата и легко обнаруживается при помощи операции «исключающее ИЛИ» над старшим битом результата и флагом переноса. Большинство процессоров осуществляют эту операцию аппаратно и помещают результат во

флаг переполнения.

6) Просуммируем числа 100 и 31

$$\begin{array}{r} 110011100 \\ +\,111100001 \\ \hline 101111101 \end{array} \Leftrightarrow \begin{array}{r} -100 \\ +\,-31 \\ \hline -131!! \end{array}$$

В этом примере в результате операции над отрицательными числами при суммировании произошло переполнение восьмибитовой переменной, т.к. получился положительный результат. И в этом случае, если рассмотреть флаг переноса, то он не совпадает со знаком результата. Отличие от предыдущего случая только в комбинации этих бит. В примере 5 говорят о переполнении результата (комбинация 01), а в примере 6 об антипереполнении результата (комбинация 10).

Умножение двоичных чисел.

Пример. Умножить числа $1101_{(2)}$ и $101_{(2)}$.

Множимое	1101	13
	X	X
Множитель	101	5
Частичное произведение	1101	65_{10}
Частичное произведение	0000	
Частичное произведение	1101	
Конечное произведение	1000001_2	

Кодирование вещественных чисел.

Иной способ применяется для представления в памяти персонального компьютера действительных чисел. Рассмотрим представление вещественных чисел с плавающей точкой.

Любое действительное число можно записать в стандартном виде $M \cdot 10^p$, где $1 \le M < 10$, p — целое. Например, $130900000 = 1{,}309 \cdot 10^8$. Поскольку каждая позиция десятичного числа отличается от соседней на степень числа 10, умножение на 10 эквивалентно сдвигу десятичной запятой на одну позицию вправо. Аналогично деление на 10 сдвигает десятичную запятую на позицию влево. Поэтому приведенный выше пример можно продолжить: $130900000 = 1{,}309 \cdot 10^8 = 0{,}1309 \cdot 10^9 = 13{,}09 \cdot 10^7$. Десятичная запятая «плавает» в числе и больше не помечает абсолютное место между целой и дробной частями.

В приведенной выше записи M называют мантиссой числа, а p – его порядком. Для того чтобы сохранить максимальную

точность, вычислительные машины почти всегда хранят мантиссу в нормализованном виде. Мантисса в таком представлении должна удовлетворять условию: $0{,}1s < m < s$. Иначе говоря, мантисса меньше 1 и первая значащая цифра — не ноль (s — основание системы счисления). При таком представлении запятая будет расположена в мантиссе перед первой значащей цифрой, что при фиксированном количестве разрядов, отведенных под мантиссу, обеспечивает запись максимального количества значащих цифр числа, то есть максимальную точность представления числа в компьютере.

Примеры нормализованного представления чисел:

$879{,}45 = 0{,}87945 \cdot 10^5$, $-0{,}0024 = 0{,}24 \cdot 10^{-2}$.

Способ хранения мантиссы с плавающей точкой подразумевает, что двоичная запятая находится на фиксированном месте. Фактически подразумевается, что двоичная запятая следует после первой двоичной цифры, т.е. нормализация мантиссы делает единичным первый бит, помещая тем самым значение между единицей и двойкой. Место, отводимое для числа с плавающей точкой, делится на два поля. Одно поле содержит знак и значение мантиссы, а другое содержит знак и значение порядка.

В стандарте IEEE 754 описан стандарт представления чисел с одинарной точностью (float) и с двойной точностью (double). Для записи числа в формате с плавающей запятой одинарной точности требуется тридцатидвухбитовое слово. Для записи чисел с двойной точностью требуется шестидесятичетырехбитовое слово. Чаще всего числа хранятся в нескольких соседних ячейках памяти процессора. Форматы числа в формате с плавающей запятой одинарной точности и числа в формате с плавающей запятой удвоенной точности приведены на рис. 7.

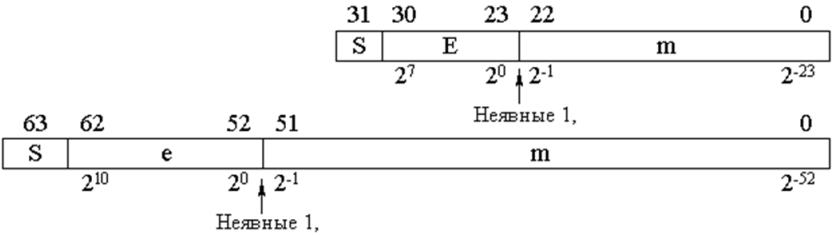

Рис. 7. Форматы числа в формате с плавающей запятой

На рисунке буквой S обозначен знак числа, 0 – это положительное число, 1 – отрицательное число. E обозначает смещённый порядок числа. Смещение требуется, чтобы не вводить в

число еще один знак. Смещённый порядок всегда положительное число. Для одинарной точности для порядка выделено восемь бит. Для смещённого порядка двойной точности отводится 11 бит. Для одинарной точности смещение принято 127, а для двойной точности – 1023. В десятичной мантиссе после запятой могут присутствовать цифры 1-9, а в двоичной – только 1. Поэтому для хранения единицы после двоичной запятой не выделяется отдельный бит в числе с плавающей запятой. Единица подразумевается, как и двоичная запятая. Кроме того, в формате чисел с плавающей запятой принято, что мантисса всегда больше 1. То есть диапазон значений мантиссы лежит в диапазоне от 1 до 2.

Алгоритм преобразования чисел в тип Double.

Покажем преобразование действительного числа для представления его в памяти ЭВМ на примере величины типа Double.

Как видно из табл. 1.2, величина типа Double занимает в памяти 8 байт. На рис. 8 показано, как здесь представлены поля мантиссы и порядка (нумерация битов осуществляется справа налево).

Знак числа	Смещенный порядок					Мантисса			
63	62	61	60	...	52	51	50	...	0

Рис. 8. Представление поля мантиссы и порядка величины типа Double

Можно заметить, что старший бит, отведенный под мантиссу, имеет номер 51, т.е. мантисса занимает младшие 52 бита. Черта указывает здесь на положение двоичной запятой. Перед запятой должен стоять бит целой части мантиссы, но поскольку она всегда равна 1, здесь данный бит не требуется и соответствующий разряд отсутствует в памяти (но он подразумевается). Значение порядка хранится здесь не в дополнительном коде. Для упрощения вычислений и сравнения действительных чисел значение порядка в ЭВМ хранится в виде смещенного числа, т.е. к настоящему значению порядка перед записью его в память прибавляется смещение. Смещение выбирается так, чтобы минимальному значению порядка соответствовал нуль. Например, для типа Double порядок занимает 11 бит и имеет диапазон от 2^{-1023} до 2^{1023}, поэтому смещение равно $1023_{(10)} = 1111111111_{(2)}$.

Наконец, бит с номером 63 указывает на знак числа.

Таким образом, для получения представления действитель-

ного числа в памяти ЭВМ необходимо выполнить следующие действия:

1) перевести модуль данного числа в двоичную систему счисления;

2) нормализовать двоичное число, т.е. записать в виде $M \cdot 2^p$, где M — мантисса (ее целая часть равна $1_{(2)}$) и p — порядок, записанный в десятичной системе счисления;

3) прибавить к порядку смещение и перевести смещенный порядок в двоичную систему счисления;

4) учитывая знак заданного числа (0 — положительное; 1 — отрицательное), выписать его представление в памяти ЭВМ.

Пример. Запишем код числа –302,25. Двоичная запись модуля этого числа имеет вид 100101110,01. Имеем $100101110{,}01_{(2)} = 1{,}0010111001 \cdot 2^8$. Получаем смещенный порядок 8+1023=1031. Далее имеем $1031_{(10)} = 10000000111_{(2)}$. Окончательно код числа –302,25 представим в табл. 1.

Таблица 1. Кодирование числа –302,25

Код	1	10000000111	00101110010000000000000000000000000000000000000000000000000000
Разряд	63	62 ... 52	51 50 ... … 0

Порядок выполнения работы

1. Получить вручную представление целого положительного числа в дополнительном коде в формате int (16 бит) в соответствии с индивидуальным заданием.

2. Получить вручную представление целого отрицательного числа в прямом отрицательном, инверсном и дополнительном кодах в формате int (16 бит) в соответствии с индивидуальным заданием.

3. Сложить первое число со вторым в соответствии с правилами выполнения машинных арифметических операций. Определить возникло ли переполнение при сложении.

4. Умножить два однобайтовых двоичных числа по формуле:

(Последние 2 цифры номера зачетки+3) * (Предпоследняя цифра номера зачетки+7).

5. Получить вручную представление вещественного числа в тип double в соответствии с индивидуальным заданием.

Варианты индивидуальных заданий

Для каждого варианта дано три числа: два целых и одно

вещественное:

1) $9600, -200, -34{,}134471 \cdot 10^{3}$

2) $2128, -100, 24{,}235 \cdot 10^{-2}$

3) $202, -77, -414{,}165 \cdot 10^{-3}$

4) $35, -4122, 1{,}3442 \cdot 10^{2}$

5) $3421, -342, 10{,}41635 \cdot 10^{4}$

6) $121, -1220, -30{,}23245 \cdot 10^{2}$

7) $1233, -127, 16{,}67 \cdot 10^{-3}$

8) $765, -9342, -100{,}788 \cdot 10^{1}$

9) $-1321, 742, 8{,}912 \cdot 10^{2}$

10) $21, -1342, -2{,}4356 \cdot 10^{2}$

11) $131, -4111, -0{,}16543 \cdot 10^{2}$

12) $12342, -88, -16{,}9129 \cdot 10^{2}$

13) $500, -1942, 33{,}543 \cdot 10^{-3}$

14) $187, -1232, 138{,}555 \cdot 10^{-4}$

15) $121, -1942, -1{,}8958 \cdot 10^{2}$

Контрольные вопросы

1. Назовите виды двоичных кодов.
2. В каком диапазоне могут храниться беззнаковые числа в двухбайтовом машинном слове?
3. В каком диапазоне могут храниться знаковые числа в двухбайтовом машинном слове в дополнительном двоичном коде?
4. В каком диапазоне могут храниться знаковые числа в двухбайтовом машинном слове в прямом двоичном коде?
5. Как получить обратный двоичный код числа?
6. Как получить дополнительный двоичный код числа?

7. Как определить возникновение переполнения при сложении чисел?

8. Каков принцип кодирования вещественных чисел?

9. Каким образом осуществляется кодирование вещественных чисел двойной точности?

Список литературы

1. Айдинян А.Р. Аппаратные средства вычислительной техники. — М., Берлин: Директ-медиа, 2016.