

Арифметика и коды чисел

Прочитайте каждый раздел и сразу выполните небольшое задание в конце. Ответы вводите прямо в поля. После завершения нажмите кнопку «Отправить работу — сохранить PDF».

ФИО

Литвинов Кирилл Русланович

Группа

ИС 25/9-1

Дата

2026-09-23

1

Как компьютер представляет числа

Человек чаще использует десятичную систему счисления, а цифровая вычислительная техника работает с битами — двумя состояниями, которые условно записывают как **0** и **1**.

Например:

$13_{10} = 1101_2$
 $25_{10} = 11001_2$

Важно: одна и та же последовательность битов может означать разные значения в зависимости от того, как её интерпретировать.

11111111_2

как 8-битное беззнаковое число = 255

как 8-битное знаковое число в дополнительном коде = -1

Поэтому процессору недостаточно просто получить набор битов: нужно знать формат данных и разрядность числа.

Задание 1

Переведите число 10_{10} в двоичную систему.

Ваш ответ

Чтобы перевести десятичную в двоичную надо число делить на 2, без остатка - 0, с остатком - 1

$$10:2=5 \text{ (0)}$$

$$5:2=2 \text{ (1)}$$

$$2:2=1 \text{ (0)}$$

$$1:2=0 \text{ (1)}$$

Потом пишем цифры наоборот, было 0101, а ответ 1010.

2

Двоичное сложение

Двоичные числа складываются по тем же разрядам, что и десятичные, но возможны только цифры 0 и 1.

$$0 + 0 = 0$$

$$0 + 1 = 1$$

$$1 + 0 = 1$$

$$1 + 1 = 10$$

$$1 + 1 + 1 = 11$$

Если получается 10_2 , в текущем разряде записывается 0, а 1 переносится в следующий разряд.

$$\begin{array}{r} 1011 \\ + 0110 \\ \hline 10001 \end{array}$$

Проверка: $1011_2 = 11_{10}$, $0110_2 = 6_{10}$, поэтому результат $10001_2 = 17_{10}$.

Задание 2

Выполните сложение: $1010_2 + 0101_2$.

Ваш ответ

$$1010 + 0101 = 1111$$

$$\text{тк } 1+0 = 1, 0+1 = 1, 1+0 = 1, 0+1 = 1$$

Проверка: 1010 в дв = $0 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 = 2$ в дес и тд., 0101 в дв = $0 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 5$ в дес, $2+5 = 7$, 1111 в дв = $1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 15$

3

Двоичное вычитание

При вычитании действуют простые правила:

$$\begin{aligned} 0 - 0 &= 0 \\ 1 - 0 &= 1 \\ 1 - 1 &= 0 \\ 10 - 1 &= 1 \end{aligned}$$

Если в текущем разряде единицы недостаточно, выполняется заём из старшего разряда.

$$\begin{array}{r} 10110 \\ - 00111 \\ \hline 01111 \end{array}$$

Проверка: $22 - 7 = 15$.

В процессоре вычитание часто удобно свести к сложению:

$$A - B = A + (-B)$$

Для этого отрицательное число представляют специальным кодом.

Задание 3

Вычислите: $1010_2 - 0011_2$.

Ваш ответ

1010

-0011 =

0111, тк занимаем справа, получается 1, потом занимаем с самого первого, у нас получается $10-1 = 1$, $1-0 = 1$, $0-0 = 1$ тк занимали с самого левого.

$1010 = 10$, $0011 = 1*2$ в $0+1*2$ в $1 = 1+2 = 3$

$10-3 = 7$

$0111 = 1+2+1*2$ в $2 = 3+4 = 7$

0111

4

Двоичное умножение

Умножение выполняется аналогично привычному умножению столбиком.

```
0 × 0 = 0
0 × 1 = 0
1 × 0 = 0
1 × 1 = 1
```

Пример:

```
  1011
× 0101
-----
  1011
 0000
 1011
0000
-----
110111
```

$1011_2 = 11$, $0101_2 = 5$, а $110111_2 = 55$.

Сдвиг двоичного беззнакового числа влево на один разряд соответствует умножению на 2, если результат помещается в доступную разрядность.

Задание 4

Умножьте: $101_2 \times 10_2$.

Ваш ответ

101
x10 =
000
101 =
1010 в дв

5

Деление и двоичные сдвиги

Для беззнаковых двоичных чисел сдвиг вправо на один разряд соответствует целочисленному делению на 2.

$$11010_2 = 26_{10}$$

$$11010 \gg 1 = 01101_2$$

$$01101_2 = 13_{10}$$

При логическом сдвиге вправо освободившиеся старшие разряды заполняются нулями. Для знаковых чисел процессор может использовать арифметический сдвиг, который сохраняет знак числа.

Задание 5

Выполните логический сдвиг вправо на один разряд: $10100_2 \gg 1$.

Ваш ответ

$10100 \gg 1 = 01010 = 1010$
 $10100 = 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0 = 4 + 16 = 20$
 $1010 = 10$ тк уже считали

6

Зачем нужны коды отрицательных чисел

Для положительного числа достаточно обычной двоичной записи:

$+13 = 00001101$

Но память компьютера не хранит отдельный символ «минус». В регистре всё равно находятся только биты. Поэтому нужно правило, по которому определённая комбинация битов будет означать отрицательное число.

При изучении архитектуры ЭВМ рассматривают три варианта:

- прямой код;
- обратный код;
- дополнительный код.

В современных процессорах для знаковых целых чисел основное значение имеет дополнительный код.

Задание 6

Объясните одним-двумя предложениями, зачем компьютеру нужен специальный способ записи отрицательных чисел.

Ваш ответ

Это сделано затем, чтобы не усложнять вычисления добавляя отдельное действие, а свести вычитание к сложению с отрицательным числом.

7

Прямой код

В прямом коде старший бит используется как знак:

0xxxxxxx — положительное число
1xxxxxxx — отрицательное число

Например, в 8-разрядной записи:

$+13 = 00001101$
 $-13 = 10001101$

Оставшиеся семь битов содержат модуль числа. Недостаток такого способа — два варианта нуля:

00000000 = +0
10000000 = -0

Кроме того, арифметические схемы должны отдельно учитывать знак операндов.

Задание 7

Запишите **-5** в 8-битном прямом коде.

Ваш ответ

5 = 5:2 = 2 (1) 2:2=1 (0), 1:2 = 0 (1) = 101
-5 = 10000101 тк 1 в начале это минус

8

Обратный код

Положительные числа записываются как обычно. Для получения отрицательного числа все биты положительного представления инвертируются.

+13 = 00001101

инверсия:
11110010

-13 в обратном коде = 11110010

У обратного кода также существуют два нуля:

00000000 = +0
11111111 = -0

Задание 8

Получите 8-битный обратный код числа **-5**.

Ваш ответ

5 = 00000101
обратный -5 = 1111010

9

Дополнительный код

Дополнительный код — основной способ представления знаковых целых чисел в современных двоичных вычислительных системах.

Чтобы получить отрицательное число:

1. запишите положительное число в нужной разрядности;
2. инвертируйте все биты;
3. прибавьте 1.

Пример для **-13**:

+13	00001101
инверсия	11110010
+ 1	00000001

-13	11110011

Короткое правило: дополнительный код отрицательного числа = инверсия битов модуля + 1.

Задание 9

Запишите **-12** в 8-битном дополнительном коде.

Ваш ответ

12 = 12:2 = 6(0), 6:2 = 3(0), 3:2 = 1(1), 1:2=0(1) = 1100 те 00001100
инверсия - 11110011
+ 00000001
= 11110100

10

Арифметика в дополнительном коде

Главное преимущество дополнительного кода — положительные и отрицательные числа можно складывать одной и той же двоичной схемой.

Рассмотрим **7 + (-5)**.

+7 = 00000111

+5	00000101
инверсия	11111010
+1	11111011
-5	11111011

```

00000111
+ 11111011
-----
1 0000010

```

При 8-битной арифметике крайний перенос за пределы разрядной сетки отбрасывается. Остаётся $0000010_2 = 2$.

Задание 10

Используя 8-битный дополнительный код, вычислите: $5 + (-3)$.

Ваш ответ

```

5 = 00000101, 3 = 00000011
3 инверсия = 11111100
+1 = 11111101
00000101+
11111101 = 1 00000010, но 1 в начале убираем тк не нужно
10 = 2
5-3 = 2

```

11

Как прочитать отрицательное число в дополнительном коде

Если старший бит знакового числа равен 1, число отрицательное. Чтобы определить его модуль, можно выполнить обратное преобразование: инвертировать биты и прибавить 1.

11111011

инверсия:
00000100

+1:
00000101 = 5

значит:
11111011 = -5

Задание 11

Определите десятичное значение 8-битного числа **11111110** в дополнительном коде.

Ваш ответ

11111110
инверсия 00000001
+ 1 = 00000010 = 2
значит 11111110 = -2

12

Диапазон представимых чисел

Количество битов определяет, какие значения можно хранить.

Для беззнакового **n-битного** числа:

0 ... 2ⁿ - 1

Для знакового числа в дополнительном коде:

-2ⁿ⁻¹ ... 2ⁿ⁻¹ - 1

Разрядность	Без знака	Со знаком, дополнительный код
8 бит	0 ... 255	-128 ... 127
16 бит	0 ... 65 535	-32 768 ... 32 767
32 бита	0 ... 4 294 967 295	-2 147 483 648 ... 2 147 483 647

Задание 12

Укажите диапазон **4-битного беззнакового** числа.

Ваш ответ

0 ... 2 в 4 -1 = 0 ... 15

13

Арифметическое переполнение

Разрядность регистра ограничена. Если математический результат нельзя представить в заданном количестве битов, возникает переполнение.

В 8-битном дополнительном коде максимальное положительное число — 127:

01111111 = 127

```
  01111111
+ 00000001
-----
 10000000
```

Но 10000000 в 8-битном дополнительном коде означает -128. Математический результат 128 не помещается в диапазон -128 ... 127.

Задание 13

Будет ли переполнение при сложении двух 8-битных знаковых чисел: **100 + 20**?

Ваш ответ

нет, тк $120 < 127$ значит входит в диапазон

14

Перенос и переполнение — разные события

При беззнаковой арифметике важен перенос из старшего разряда. Например:

```
 11111111
+ 00000001
-----
1 00000000
```

В 8 битах останется `00000000`, а единица окажется за пределами регистра. Это перенос.

Для знаковой арифметики в дополнительном коде отдельно отслеживается переполнение — ситуация, когда знаковый математический результат не помещается в заданную разрядность.

В архитектуре x86 эти ситуации отражаются разными флагами: CF связан с переносом/заёмом для беззнаковой арифметики, а OF — с переполнением знаковой арифметики.

Задание 14

Какой флаг в x86 связан со знаковым переполнением: **CF** или **OF**?

Ваш ответ

OF — с переполнением знаковой арифметики.

15

Расширение знака

Иногда число нужно перенести из регистра меньшей разрядности в регистр большей разрядности.

Для положительного числа старшие разряды заполняются нулями:

```
8 бит:   00000101
16 бит:  00000000 00000101
```

Для отрицательного числа в дополнительном коде копируется знаковый бит:

```
-5, 8 бит:
11111011

-5, 16 бит:
11111111 11111011
```

Так сохраняется числовое значение.

Задание 15

Расширьте 8-битное число **11111110** до 16 бит с сохранением знака.

Ваш ответ

11111111 11111110, единицы в начале чтобы сохранить знак

16

Двоично-десятичный код BCD

В BCD каждая десятичная цифра кодируется отдельно четырьмя битами.

5 → 0101

9 → 1001

59 в BCD:

0101 1001

Это не то же самое, что обычная двоичная запись числа 59:

$59_{10} = 00111011_2$

59 в BCD = 0101 1001

BCD удобен там, где важно работать именно с десятичными цифрами.

Задание 16

Запишите число **42** в BCD.

Ваш ответ

$4 = 4:2 = 2(0) \quad 2:2 = 1(0) \quad 1:2 = 0(1) = 100$

$2 = 10$

т.е $4 = 0100, 2 = 0010$

42 в bcd = 0100 0010

17

Целые и вещественные числа

Целые числа могут интерпретироваться как беззнаковые или знаковые. Для знаковых целых чисел широко используется дополнительный код.

Но дробные и очень большие или очень маленькие значения удобно представлять с плавающей точкой. В распространённом двоичном формате число разбивается на части:

- знак;
- порядок;
- значащая часть.

Например, стандарт IEEE 754 описывает форматы двоичной арифметики с плавающей точкой. Подробное устройство таких чисел рассматривается отдельно в теме о формах представления чисел в ЭВМ.

Задание 17

Что лучше подходит для хранения значения **3,14159**: целое число или число с плавающей точкой?

Ваш ответ

с плавающей точкой тк если сохранить как 3 то теряется большое колво инфы

18

АЛУ и флаги результата

Арифметико-логическое устройство (АЛУ) процессора выполняет арифметические и логические операции над двоичными операндами.

```
5 = 00000101
3 = 00000011

00000101 + 00000011
    ↓
  00001000
    ↓
    8
```

После операции процессор может изменить флаги состояния. В x86 среди них:

- **CF** — перенос/заём;
- **ZF** — результат равен нулю;
- **SF** — знак результата;
- **OF** — знаковое переполнение.

Эти признаки позволяют следующим машинным инструкциям анализировать результат вычисления.

Задание 18

Какой флаг устанавливается, если результат арифметической операции равен нулю?

Ваш ответ

ZF — результат равен нулю

Перед сдачей

- ✓ Указаны ФИО и группа.
- ✓ Заполнены ответы после каждого раздела.
- ✓ В вычислениях указаны промежуточные действия там, где они требуются.

После нажатия кнопки откроется окно печати. Выберите «Сохранить как PDF». В PDF попадут все ваши ответы полностью.