Лекция 2 *Операции над кодами чисел*

Цель лекции:

Рассмотреть систематизированные основы знаний по кодированию числовой информации и выполнению операций над кодами чисел

Учебные вопросы:

1. Представление целых беззнаковых чисел
2. Представление чисел с фиксированной точкой
3. Операции над числами с фиксированной точкой
4. Представление чисел с плавающей точкой
5. Арифметические операции над числами с плавающей точкой

1. Представление целых беззнаковых чисел

Структура памяти

- Память состоит из **нумерованных ячеек**.
- **Линейная структура** (адрес ячейки – одно число).
- **Байт** – это наименьшая ячейка памяти, имеющая собственный адрес.

На современных компьютерах **1 байт = 8 бит**.

Целые беззнаковые числа

Беззнаковые данные – не могут быть отрицательными.

Для представления беззнаковых чисел может использоваться 1 или несколько байт.

Для беззнаковых в **1 байт (8 бит)** диапазон значений

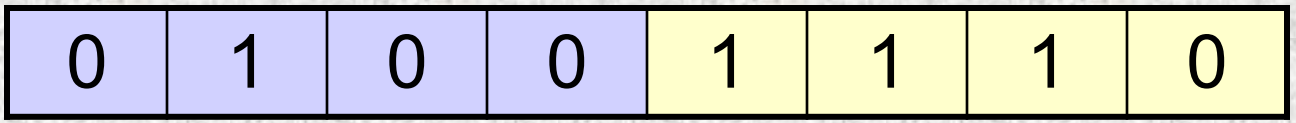
0…255, **$0…FF_{16} = 2^8 – 1$.**

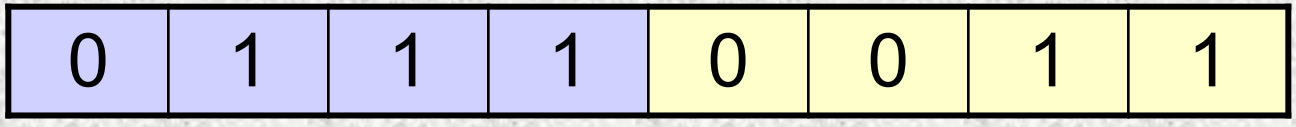
Си: ***unsigned char*** **Паскаль:** ***byte***

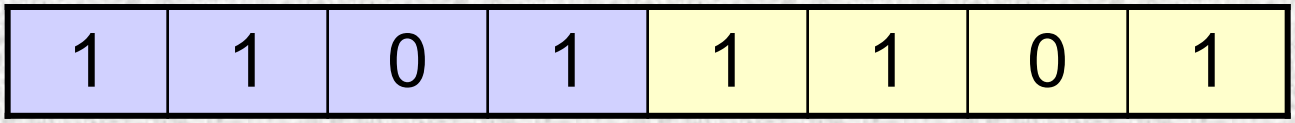
4_{16} E_{16}

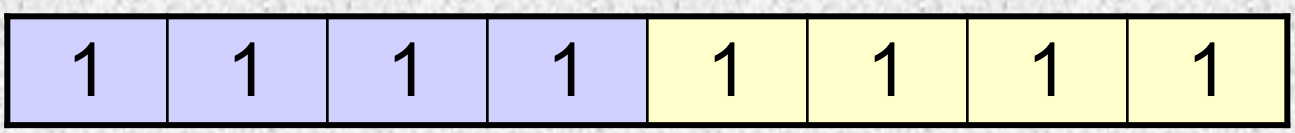
$1001110_2 = 4E_{16} =$ ‘N’

Примеры

78 = 

115 = 

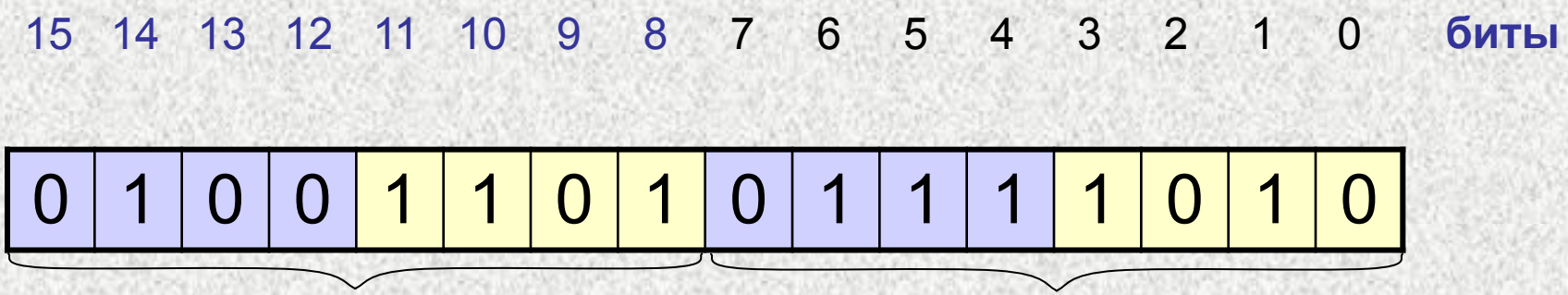
221 = 

255 = 

Целое без знака размером 2 байта (16 бит)

диапазон значений **0…65535, 0…FFFF$_{16}$ = 2^{16}-1**

Си: ***unsigned int*** **Паскаль:** ***word***



100110101111010_2 = $4D7A_{16}$

Длинное целое без знака рамером 4 байта

диапазон значений **0…FFFFFFFF$_{16}$ = 2^{32}-1**

Си: ***unsigned long int*** **Паскаль:** ***dword***

2. Представление чисел с фиксированной точкой

Числа с фиксированной точкой — форма представления вещественных чисел, когда точкой отделяется целая часть от дробной. Например, 12.54

С фиксированной точкой все числа изображаются в виде последовательности цифр с постоянным для всех чисел положением точки, отделяющей целую часть числа от дробной.

Число в компьютере занимает определенное количество разрядов (бит), кратное 8:

8, **16**, **32**, **64**.

Пример 1.

Пусть даны числа **123.6**, **0.01** и **1.5143**.

Записать их в виде числа с фиксированной точкой при условии, что под каждое число выделяется 8 разрядов – 4 под целую часть и 4 под дробную.

Ответ:

Десятичное число	Число с фиксированной точкой
123.6	**0123.6000**
0.01	**0000.0100**
1.5143	**0001.5143**

Такая форма представления наиболее проста для восприятия, но имеет существенный недостаток:

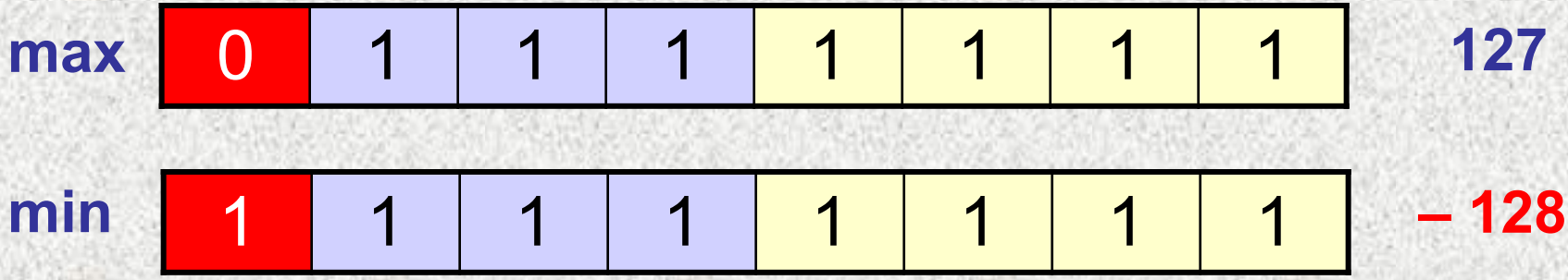
- программирование операций для чисел с фиксированной точкой, требует усилий по отслеживанию положения точки.

Поэтому в современных компьютерах форма представления чисел с фиксированной точкой используется как вспомогательная.

Числа с фиксированной точкой в ЭВМ

Старший (знаковый) бит числа определяет его знак. Если он равен **0**, число положительное, если 1, то **отрицательное**.

В восьми разрядной сетке числа с фиксированной точкой (байтовые числа) могут быть представлены в диапазоне значений:



$-2^{n-1} \ldots 2^{n-1}-1$

$-128 = -2^7 \ldots 127 = 2^7 - 1$

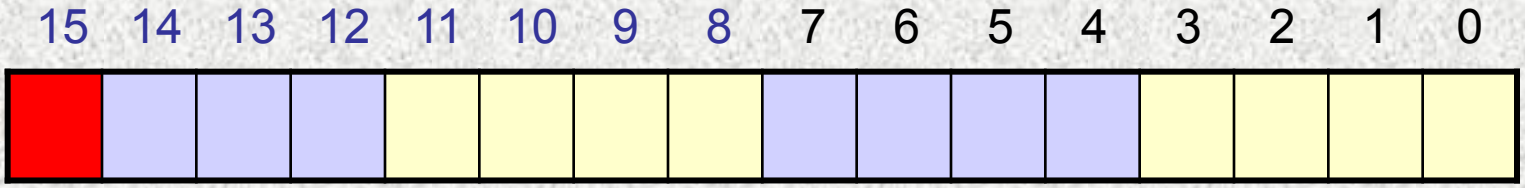
можно работать с отрицательными числами

уменьшился диапазон положительных чисел

Слово со знаком занимает 2 байта (16 бит) и может быть представлено в диапазоне значений

$$-2^{15} \ldots 2^{15-1}$$

$$-32768 \ldots 32767$$



Си: ***int*** **Паскаль:** ***integer***

Двойное слово со знаком занимает 4 байта и диапазон значений

$$-2^{31} \ldots 2^{31}-1$$

Си: ***long int*** **Паскаль:** ***longint***

3. Коды чисел и операции над ними

3. Коды чисел и операции над ними

3.1 Прямой код

Такой способ представления числа является наиболее естественным, так как обеспечивает простой переход от двоичной записи числа к записи его двоичного (прямого) кода:

$$X_{10} \Rightarrow X_2 \Rightarrow X_{ПК}$$

$$-X_{10} \Rightarrow -X_2 \Rightarrow X_{ПК}$$

Прямой код имеет существенные недостатки:

- использовании **прямого кода** имеется целых два кода для представления **0**

  +0 представляется как **0000 0000**

  -0 представляется как **1000 0000**;

- сложно реализовать операцию вычитания, умножения и деления чисел.

3.2 Обратный код

Обратный код для положительных чисел совпадает с прямым кодом,

$$X_{ОК} = X_{ПК}$$

Обратный код для отрицательных чисел получается инвертированием всех цифр двоичного кода абсолютной величины числа.

$$-X_{10} \Rightarrow -X_2 \Rightarrow X_{ПК} \Rightarrow \overline{X}_{ОК}$$

где - $\overline{X}$ операция инвертирования двоичного кода абсолютной величины числа.

Пример представления числа -123_{10} в обратном коде:

1. Перевод числа -123_{10} в двоичное число:

$$-123_{10} \Rightarrow -1111011_2$$

2. Запись двоичного числа в прямом коде:

$$-1111011_2 \Rightarrow 11111011_{ПК}$$

3.Перевод прямого кода двоичного числа в обратный:

$$11111011_{ПК} \Rightarrow 10000100$$

Для обратного кода справедливо следующее соотношение:

$$X_{10} = x_{sign} * (-2^{n-1} + 1) + \sum_{i=0}^{n-2} x_i * 2^i$$

где

x_{sign} – значение знакового разряда,

x_i – значение i-го разряда,

n – разрядность числа.

Например:

$1010 = 1*(-2^3+1)+0*2^0+1*2^1+0*2^2 = -7+2 = -5$

При использовании обратного кода арифметические операции для отрицательных чисел выполняются проще.

Само получение обратных кодов не требует значительных усилий.

Но, как и в случае прямого кода, 0 представляется 2 кодами:

+0 представляется как **0000 0000**

-0 представляется как **1111 1111**

3.3 Дополнительный код

Дополнительный код для положительных чисел совпадает с прямым кодом,

$$X_{ОК} = X_{ПК}$$

Дополнительный код для отрицательных чисел получается прибавлением к младшему разряду обратного кода числа 1.

$$X_{ПК} = X_{ОК} + 1$$

Пример представления числа -115_{10} в дополнительном коде:

1. Перевод числа -115_{10} в двоичное число:

$$-115_{10} \Rightarrow -1110011_2$$

2. Запись двоичного числа в прямом коде:

$$-1111011_2 \Rightarrow 11110011_{ПК}$$

3. Перевод прямого кода двоичного числа в обратный:

$$11111011_{ПК} \Rightarrow 10001100_{ОК}$$

4. Перевод обратного кода двоичного числа в дополнительный:

$$\begin{array}{r} 10001100_{ОК} \\ +\ 00000001 \\ \hline 10001101_{ДК} \end{array}$$

Для дополнительного кода справедливо следующее соотношение:

$$X = x_{sign} * (-2^{n-1}) + \sum_{i=0}^{n-2} x_i * 2^i$$

где

x_{sign} – значение знакового разряда,
x_i – значение i-го разряда,
n – разрядность числа.

Например:

$$1011 = 1*(-2^3)+1*2^0+1*2^1+0*2^2 = -8+3 = -5$$

Анализ дополнительного кода.

- Получение дополнительного кода происходит несколько сложнее, чем обратного.

- Использование дополнительного кода упрощает арифметические операции, что будет показано позже.

- В дополнительном коде **0** имеет единственное представление :

$$+0=0000\ 0000_{ДК}$$

$$-0=1111\ 1111_{ОК}+1=0000\ 0000_{ДК}$$

- Использование дополнительного кода дает возможность расширить диапазон представления чисел (закодировать 256 чисел) по сравнению с прямым и обратным кодами (255 чисел).

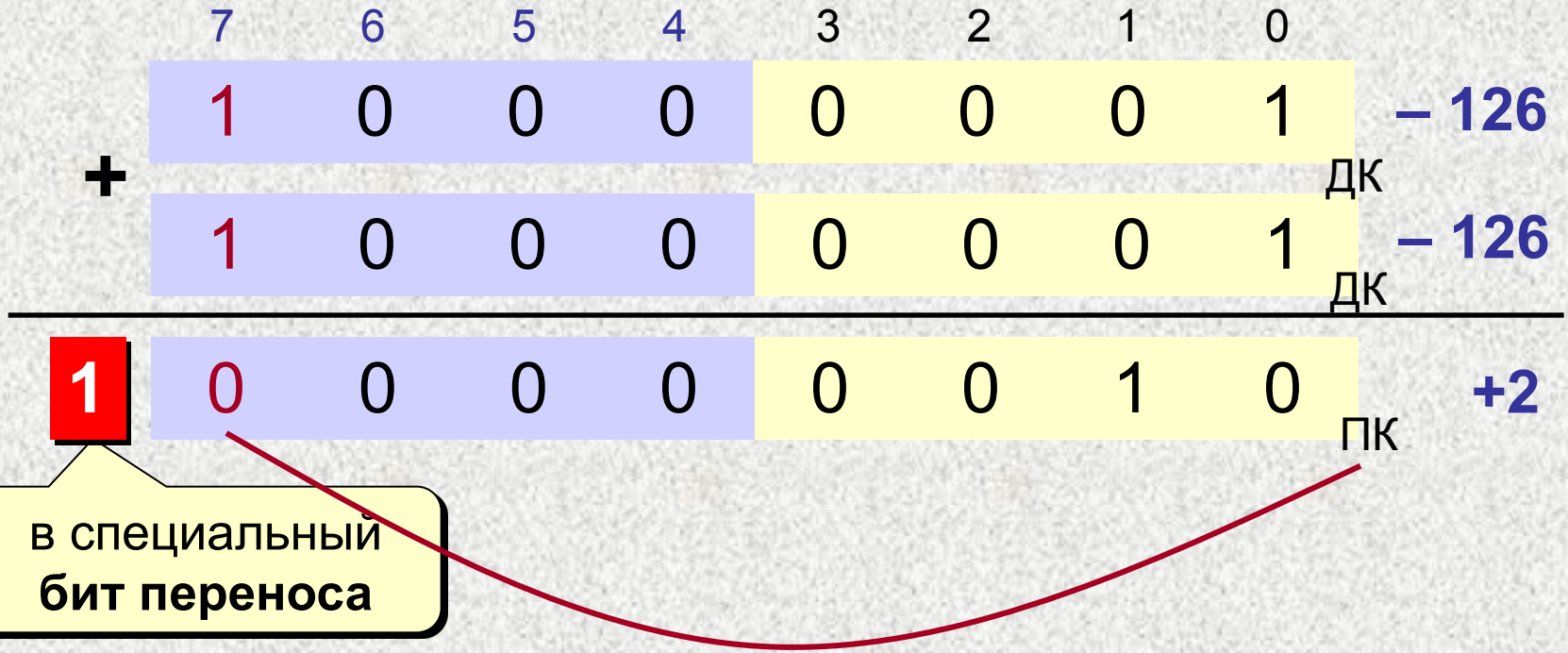
Последнее утверждение доказать самостоятельно!

3.4 Ошибки выполнения операций над кодами

Переполнение разрядной сетки: в результате сложения больших положительных чисел получается отрицательное (**перенос в знаковый бит**).

	7	6	5	4	3	2	1	0	
	0	1	0	0	0	0	0	0	64
+	0	1	0	0	0	0	0	0	64
	1	0	0	0	0	0	0	0	– 0

Перенос: при сложении больших (по модулю) отрицательных чисел получается положительное (перенос **за границы разрядной сетки**).



Учебные вопросы для самостоятельного изучения

4. Представление чисел с плавающей точкой

$$X = s \cdot M \cdot 2^e$$

s – **знак** (1 или -1)

M – **мантисса**, $M = 0$ **или** $1 \le M < 2$

e – **порядок**

Пример:

знак

мантисса

порядок

$$15{,}625 = 1111{,}101_2 = 1 \cdot 1{,}111101_2 \cdot 2^3$$

$$3{,}375 =$$

Нормализованные числа в памяти

IEEE Standard for Binary Floating-Point Arithmetic (IEEE 754)

$$15{,}625 = 1 \cdot 1{,}111101_2 \cdot 2^3$$

$s = 1$ $e = 3$ $M = 1{,}111101_2$

Порядок со сдвигом:
$p = e + E$ (сдвиг)

Дробная часть мантиссы:
$m = M - 1$

Знаковый бит:
0, если s = 1
1, если s = – 1

Целая часть M всегда 1, поэтому не хранится в памяти!

Нормализованные числа в памяти

Тип данных	Размер, байт	Мантисса, бит	Порядок, бит	Сдвиг порядка, **E**	Диапазон модулей	Точность, десятичн. цифр
float **single**	**4**	23	8	127	$3{,}4\cdot10^{-38}$ … $3{,}4\cdot10^{38}$	**7**
double **double**	**8**	52	11	1023	$1{,}7\cdot10^{-308}$ … $1{,}7\cdot10^{308}$	**15**
long double **extended**	**10**	64	15	16383	$3{,}4\cdot10^{-4932}$ … $3{,}4\cdot10^{4932}$	**19**

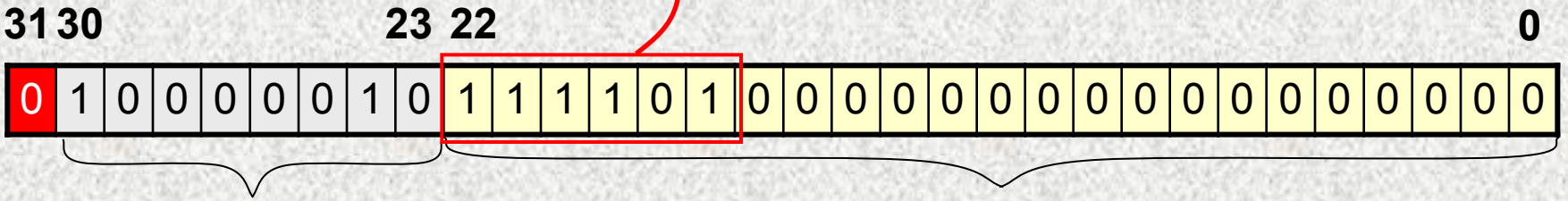
Типы данных для языков: **Си**

Паскаль

Вещественные числа в памяти

$15{,}625 = 1{,}111101_2 \cdot 2^3$

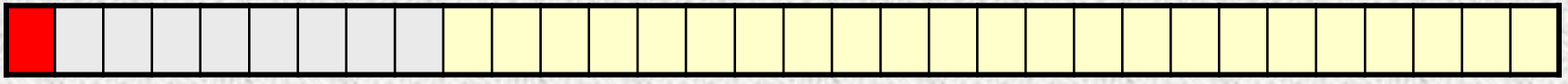
4 байта = 32 бита



$p = e + 127 = 130 = 10000010_2$

$m = M - 1 = 0{,}111101_2$

$3{,}375 =$



5. Арифметические операции над числами с плавающей точкой

сложение

$$5{,}5 + 3 = 101{,}1_2 + 11_2 = 8{,}5 = 1000{,}1_2$$

1. Порядок выравнивается до большего

$$5{,}5 = 1{,}011_2 \cdot 2^2$$

$$3 = 1{,}1_2 \cdot 2^1 = 0{,}11_2 \cdot 2^2$$

2. Мантиссы складываются

$$\begin{array}{r} 1{,}011_2 \\ +\ 0{,}110_2 \\ \hline 10{,}001_2 \end{array}$$

3. Результат нормализуется (с учетом порядка)

$$10{,}001_2 \cdot 2^2 = 1{,}0001_2 \cdot 2^3 = 1000{,}1_2 = 8{,}5$$

вычитание

$$10{,}75 - 5{,}25 = 1010{,}11_2 - 101{,}01_2 = 101{,}1_2 = 5{,}5$$

1. Порядок выравнивается до большего

$$10{,}75 = 1{,}01011_2 \cdot 2^3$$

$$5{,}25 = 1{,}0101_2 \cdot 2^2 = 0{,}10101_2 \cdot 2^3$$

2. Мантиссы вычитаются

$$\begin{array}{r} 1{,}01011_2 \\ -\ 0{,}10101_2 \\ \hline 0{,}10110_2 \end{array}$$

3. Результат нормализуется (с учетом порядка)

$$0{,}1011_2 \cdot 2^3 = 1{,}011_2 \cdot 2^2 = 101{,}1_2 = 5{,}5$$

умножение

$$7 \cdot 3 = 111_2 \cdot 11_2 = 10101_2 = 21$$

1. Мантиссы умножаются

$$7 = 1{,}11_2 \cdot 2^2$$

$$3 = 1{,}1_2 \cdot 2^1$$

$$\begin{array}{r} 1{,}1\,1_2 \\ \times\ \ 1{,}1_2 \\ \hline 1\,1\,1_2 \\ 1\,1\,1_2 \\ \hline 1\,0{,}1\,0\,1_2 \end{array}$$

2. Порядки складываются: $2 + 1 = 3$

3. Результат нормализуется (с учетом порядка)

$$10{,}101_2 \cdot 2^3 = 1{,}0101_2 \cdot 2^4 = 10101_2 = 21$$

деление

$17{,}25 : 3 = 10001{,}01_2 : 11_2 = 101{,}11_2 = 5{,}75$

1. Мантиссы делятся

$17{,}25 = 1{,}000101_2 \cdot 2^4$

$3 = 1{,}1_2 \cdot 2^1$

$1{,}000101_2 : 1{,}1_2 = 0{,}10111_2$

2. Порядки вычитаются: $4 - 1 = 3$

3. Результат нормализуется (с учетом порядка)

$0{,}10111_2 \cdot 2^3 = 1{,}0111_2 \cdot 2^2 = 101{,}11_2 = 5{,}75$