Представление данных в компьютере

АКОС, МФТИ

19 сентября, 2024

Как работают целые числа

$$137_{10} =$$

=	+128	+64	+32	+16	+8	+4	+2	+1	=

$$= 1\ 0\ 0\ 0\ 1\ 0\ 0\ 1_2$$

Целочисленные типы в C

- `char` : **1 байт** (или `CHAR_BIT` бит) данных;
- `short` и `int` : не менее **16 бит** данных;
- `long` : не менее **32 бит** данных;
- `long long` : не менее **64 бит** данных.

Типы фиксированной длины (`#include <stdint.h>`):

- `int8_t` и `uint8_t` : строго **8 бит**;
- `int16_t` и `uint16_t` : строго **16 бит**;
- `int32_t` и `uint32_t` : строго **32 бита**;
- `int64_t` и `uint64_t` : строго **64 бита**.

Как работают знаковые числа

$$-23_{10} =$$

-128	+64	+32	+16	+8	+4	+2	+1

$$= 1\ 1\ 1\ 0\ 1\ 0\ 0\ 1_2$$

⚠️ : Любое отрицательное число начинается с 1 и наоборот;

⚠️ : Конвертация знаковых типов друг к другу становится менее тривиальным.

Знаковые и беззнаковые типы в C

- `char` : **не определено стандартом**.
- `short` , `int` , `long` и `long long` : по умолчанию **знаковые**;
- Любой из типов выше можно сделать:
  - **Знаковым** (напр., `signed char`);
  - **Беззнаковым** (напр., `unsigned int`).

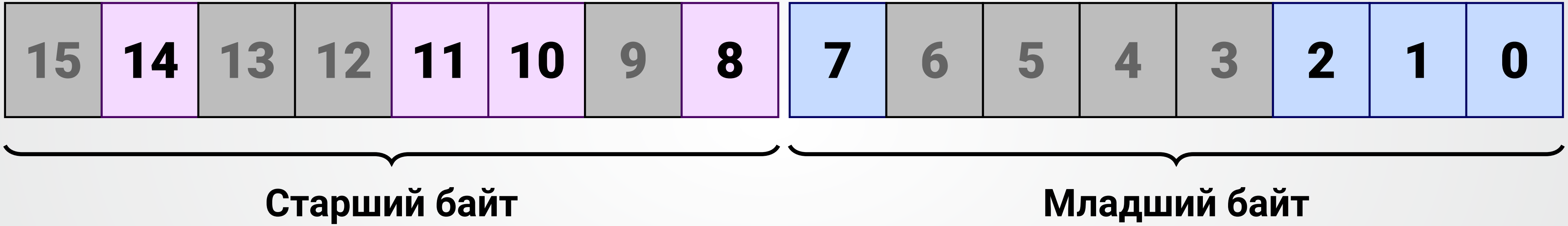
Типы фиксированной длины (`#include <stdint.h>`):

- `int16_t` : **знаковое** 16-битное число;
- `uint16_t` : **беззнаковое** 16-битное число (`u` в начале от слова `unsigned`);

⚠️ : Знаковые типы **нельзя переполнять** - в Си это UB. Беззнаковые – можно.

Как хранить длинные типы?

```
(uint16_t) 19847 =
```

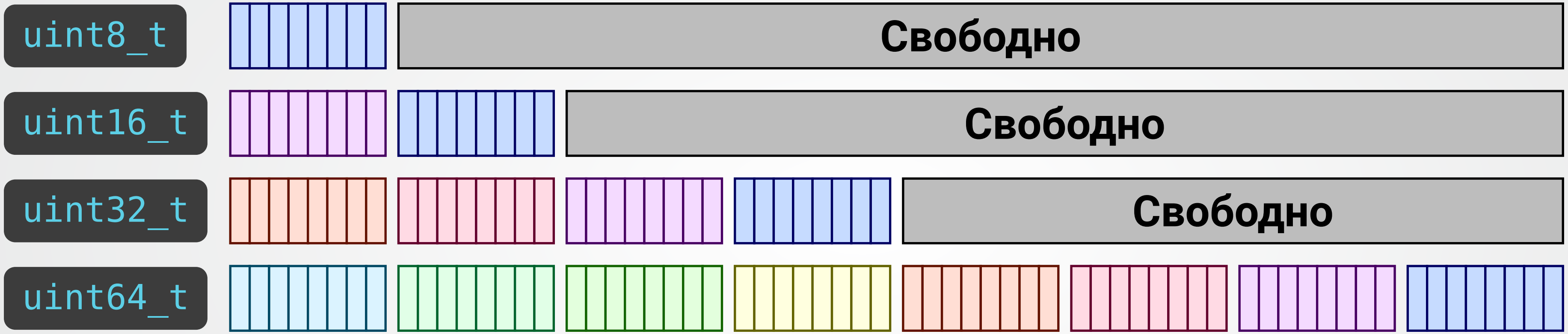


Не будет ли проблем с такой схемой?...

Что может пойти не так?

```
int main() {
  uint64_t my_long = 42;

  printf("%d\n", &my_long); // Что выведет?
}
```



⚠️ : **Младший** **синий** **байт оказывается в разных местах.**

Сложности приведения типов

```
uint8_t a_byte = 42;

uint16_t a_16b = a_byte; // Перенесёт 42 во второй байт
uint32_t a_32b = a_byte; // Перенесёт 42 в четвёртый байт
uint64_t a_64b = a_byte; // Перенесет 42 в восьмой байт
```

⚠️ : В схеме Big-Endian **каждый целочисленный каст** требует **менять порядок байт**:

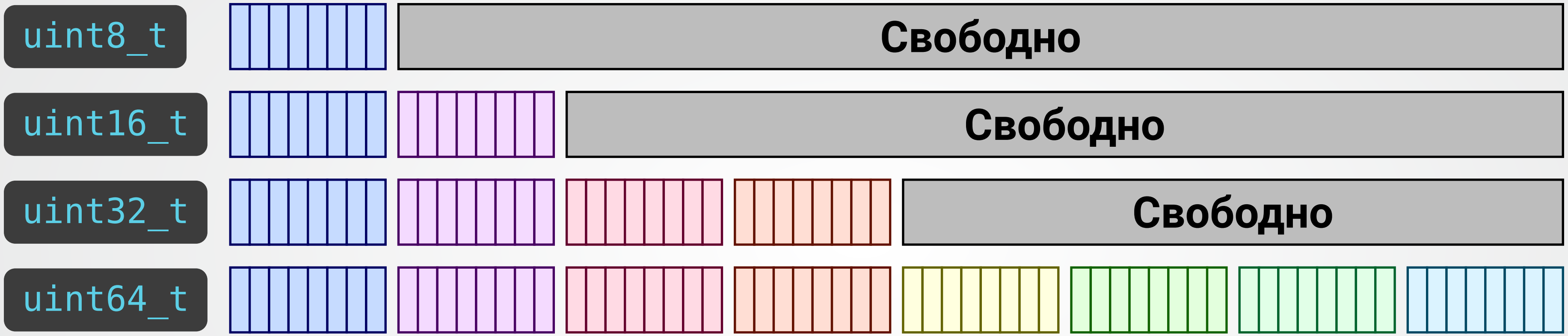
Что, если хранить байты наоборот?

`(uint16_t) 19847 =`

7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8

Младший байт (биты 7–0) **Старший байт** (биты 15–8)

Что, если хранить байты наоборот?



✓ В такой схеме (Little-Endian) приведение целочисленных типов не требует перемещения байт.

Схема Big-Endian (BE, сначала старший байт)

- \+ Удобная для человека. Порядок бит как в десятичной записи;
- \+ Позволяет ускорять `strcmp` и `memcmp` ;
- \- Иногда сложнее приводить типы.

Схема Little-Endian (LE, сначала младший байт)

- \+ Легко кастуется туда-обратно;
- \- Чуть-чуть ломает мозг.

К чему пришли люди:

- x86 всегда LE. ARM по умолчанию LE, но поддерживает оба варианта.
- В Big-Endian вводятся двоичные литералы: `0b1000000000000000 == 32768` , а не `128`
- А еще Big-Endian используется при передачи данных по сети.

Дробные числа

$$10.675_{10} =$$

=	+16	+8	+4	+2	+1	+0.5	+0.25	+0.125	=
=	0	1	0	1	0	1	0	1_2	

Несколько младших разрядов можно зарезервировать под дробную часть. Получится `fixed-point`.

Недостатки `fixed-point`

- **-** Для каждой задачи **нужно подбирать оптимальное количество дробных бит**.
- **-** При неоптимальном порядке представление либо **теряет относительную точность**, либо **быстро переполнится**.

К чему пришли люди:

- Давайте **менять число дробных бит на ходу**;
- Выделим несколько бит под **счётчик**;
- Назовём это ✨ `floating-point` ✨.

`floating-point` в IEEE 754

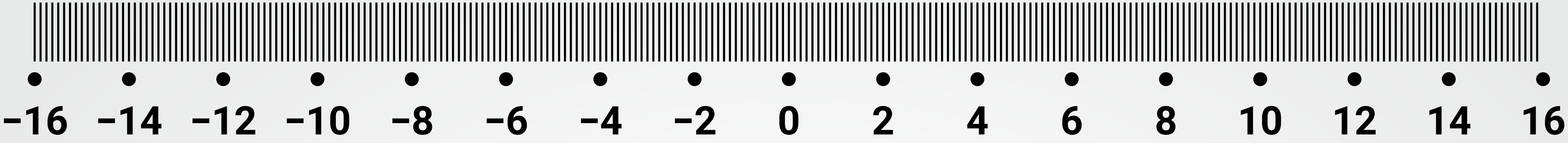
$$\pi \approx 3.125 =$$

$$= 1 \cdot -1 \cdot 2^4 \cdot 2^2 \cdot 2^1 + \tfrac{1}{2} + \tfrac{1}{4} + \tfrac{1}{8} + \tfrac{1}{16} \cdot 2^{-3} =$$

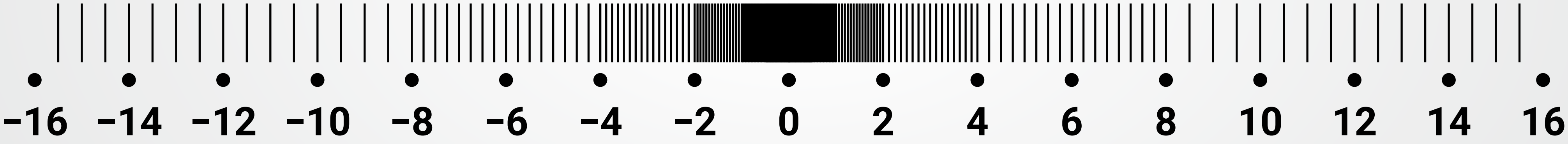
$$= 0\ 1\ 0\ 0\ 1\ 0\ 0\ 1_2$$

(Вообще, в IEEE 754 минимум 16 бит, но суть та же)

`fixed-point` (знаковый, 8 бит, 3-б. дробь)



`floating-point` (3-б. экспонента, 4-б. мантисса)



- \+ Макс. погрешность на $(1, 15)$ стала **3%** против **5.6%**;
- \- **32 из 256 значений (12%)** уходят на служебные (`NaN`, `Inf`).

Спец. значения `floating-point` в IEEE 754

`+Inf`	0	1	1	1	0	0	0	0
`-Inf`	1	1	1	1	0	0	0	0
`qNaN`	0	1	1	1	1	0	0	0
`sNaN`	0	1	1	1	0	0	0	0
`0`	0	0	0	0	0	0	0	0
`-0`	1	0	0	0	0	0	0	0

- При нулевой экспоненте включается **денормализованный режим**. Он заменяет старшую единицу на ноль. Так сохраняется плотность значений близко к нулю.

Сложение `floating-point`

$\mathrm{A} = 3.125$ $\approx \pi$

$$= -1^{\boxed{0}} \cdot 2^{\boxed{100}_2 - 3} \cdot 1.\boxed{1001}_2$$

0	1	0	0	1	0	0	1

$\mathrm{B} = 2.625$ $\approx e$

$$= -1^{\boxed{0}} \cdot 2^{\boxed{100}_2 - 3} \cdot 1.\boxed{0101}_2$$

0	1	0	0	0	1	0	1

$\mathrm{A} + \mathrm{B} =$

Сложение `floating-point`

$A = 3.125$ $\approx \pi$

$= -1^{\boxed{0}} \cdot 2^{\boxed{100}_2 - 3} \cdot 1.\boxed{1001}_2$

0	1	0	0	1	0	0	1

$B = 2.625$ $\approx e$

$= -1^{\boxed{0}} \cdot 2^{\boxed{100}_2 - 3} \cdot 1.\boxed{0101}_2$

0	1	0	0	0	1	0	1

$A + B =$ $2^{100_2 - 3} \cdot (1.1001_2 + 1.0101_2)$

Сложение `floating-point`

$A = 3.125 \approx \pi$

$= -1^{\boxed{0}} \cdot 2^{\boxed{100}_2 - 3} \cdot 1.\boxed{1001}_2$

0	1	0	0	1	0	0	1

$B = 2.625 \approx e$

$= -1^{\boxed{0}} \cdot 2^{\boxed{100}_2 - 3} \cdot 1.\boxed{0101}_2$

0	1	0	0	0	1	0	1

$A + B = 2^{100_2 - 3} \cdot (1.1001_2 + 1.0101_2) =$

$= 2^{100_2 - 3} \cdot 10.1110_2$

Сложение `floating-point`

$A = 3.125 \approx \pi$

$= -1^{\boxed{0}} \cdot 2^{\boxed{100}_2 - 3} \cdot 1.\boxed{1001}_2$

0	1	0	0	1	0	0	1

$B = 2.625 \approx e$

$= -1^{\boxed{0}} \cdot 2^{\boxed{100}_2 - 3} \cdot 1.\boxed{0101}_2$

0	1	0	0	0	1	0	1

$A + B = 2^{100_2 - 3} \cdot (1.1001_2 + 1.0101_2) =$

$= 2^{100_2 - 3} \cdot 10.1110_2 =$

$= -1^{\boxed{0}} \cdot 2^{\boxed{101}_2 - 3} \cdot 1.\boxed{0111}_2 = 5.75$

0	1	0	1	0	1	1	1

Вычитание `floating-point`

$A = 4.5$

$= -1^{\boxed{0}} \cdot 2^{\boxed{101}_2 - 3} \cdot 1.\boxed{0010}_2$

0	1	0	1	0	0	1	0

$B = -4$

$= -1^{\boxed{1}} \cdot 2^{\boxed{101}_2 - 3} \cdot 1.\boxed{0000}_2$

1	1	0	1	0	0	0	0

$A + B =$

Вычитание `floating-point`

$A = 4.5$

$= -1^{\boxed{0}} \cdot 2^{\boxed{101}_2 - 3} \cdot 1.\boxed{0010}_2$

0	1	0	1	0	0	1	0

$B = -4$

$= -1^{\boxed{1}} \cdot 2^{\boxed{101}_2 - 3} \cdot 1.\boxed{0000}_2$

1	1	0	1	0	0	0	0

$A + B =$ $2^{101_2 - 3} \cdot (1.0010_2 - 1.0000_2)$

Вычитание `floating-point`

$A = 4.5$

$= -1^{\boxed{0}} \cdot 2^{\boxed{101}_2 - 3} \cdot 1.\boxed{0010}_2$

0	1	0	1	0	0	1	0

$B = -4$

$= -1^{\boxed{1}} \cdot 2^{\boxed{101}_2 - 3} \cdot 1.\boxed{0000}_2$

1	1	0	1	0	0	0	0

$A + B =$ $2^{101_2 - 3} \cdot (1.0010_2 - 1.0000_2) \approx$

$\approx 2^{101_2 - 3} \cdot 0.0010_2$

Вычитание `floating-point`

$A = 4.5$

$= -1^{\boxed{0}} \cdot 2^{\boxed{101}_2 - 3} \cdot 1.\boxed{0010}_2$

0	1	0	1	0	0	1	0

$B = -4$

$= -1^{\boxed{1}} \cdot 2^{\boxed{101}_2 - 3} \cdot 1.\boxed{0000}_2$

1	1	0	1	0	0	0	0

$A + B =$ $2^{101_2 - 3} \cdot (1.0010_2 - 1.0000_2) \approx$

$\approx 2^{101_2 - 3} \cdot 0.0010_2 =$

$= -1^{\boxed{0}} \cdot 2^{\boxed{010}_2 - 3} \cdot 1.\boxed{0000}_2$ $= 0.5$

0	0	1	0	0	0	0	0

Сложение `floating-point` разных порядков

$\mathrm{A} = 1.625$

$$= -1^{\boxed{0}} \cdot 2^{\boxed{011}_2 - 3} \cdot 1.\boxed{1010}_2$$

0	0	1	1	1	0	1	0

$\mathrm{B} = 4.5$

$$= -1^{\boxed{0}} \cdot 2^{\boxed{101}_2 - 3} \cdot 1.\boxed{0010}_2$$

0	1	0	1	0	0	1	0

$\mathrm{A} + \mathrm{B} =$

Сложение `floating-point` разных порядков

$A = 1.625$

$= -1^{\boxed{0}} \cdot 2^{\boxed{011}_2 - 3} \cdot 1.\boxed{1010}_2$

0	0	1	1	1	0	1	0

$B = 4.5$

$= -1^{\boxed{0}} \cdot 2^{\boxed{101}_2 - 3} \cdot 1.\boxed{0010}_2$

0	1	0	1	0	0	1	0

$A + B =$ $2^{101_2 - 3} \cdot \left(1.1010_2 \cdot 2^{-2} + 1.0010_2\right)$

Сложение `floating-point` разных порядков

$A = 1.625$

$$= -1^{\boxed{0}} \cdot 2^{\boxed{011}_2 - 3} \cdot 1.\boxed{1010}_2$$

0	0	1	1	1	0	1	0

$B = 4.5$

$$= -1^{\boxed{0}} \cdot 2^{\boxed{101}_2 - 3} \cdot 1.\boxed{0010}_2$$

0	1	0	1	0	0	1	0

$A + B =$ $2^{101_2 - 3} \cdot (1.1010_2 \cdot 2^{-2} + 1.0010_2) =$

$$= 2^{101_2 - 3} \cdot (0.011010_2 + 1.0010_2)$$

Сложение `floating-point` разных порядков

$A = 1.625$

$$= -1^{\boxed{0}} \cdot 2^{\boxed{011}_2 - 3} \cdot 1.\boxed{1010}_2$$

0	0	1	1	1	0	1	0

$B = 4.5$

$$= -1^{\boxed{0}} \cdot 2^{\boxed{101}_2 - 3} \cdot 1.\boxed{0010}_2$$

0	1	0	1	0	0	1	0

$A + B =$ $2^{101_2 - 3} \cdot (1.1010_2 \cdot 2^{-2} + 1.0010_2) =$

$$= 2^{101_2 - 3} \cdot (0.011010_2 + 1.0010_2) \approx 2^{101_2 - 3} \cdot 1.1001_2$$

Сложение `floating-point` разных порядков

$A = 1.625$

$$= -1^{\boxed{0}} \cdot 2^{\boxed{011}_2 - 3} \cdot 1.\boxed{1010}_2$$

0	0	1	1	1	0	1	0

$B = 4.5$

$$= -1^{\boxed{0}} \cdot 2^{\boxed{101}_2 - 3} \cdot 1.\boxed{0010}_2$$

0	1	0	1	0	0	1	0

$$A + B = 2^{101_2 - 3} \cdot (1.1010_2 \cdot 2^{-2} + 1.0010_2) =$$

$$= 2^{101_2 - 3} \cdot (0.011010_2 + 1.0010_2) \approx 2^{101_2 - 3} \cdot 1.1001_2 =$$

$$= -1^{\boxed{0}} \cdot 2^{\boxed{101}_2 - 3} \cdot 1.\boxed{1001}_2 = 6.25$$

0	1	0	1	1	0	0	1

Денормализованные `floating-point`

$A = 0.5$

$= -1^{\boxed{0}} \cdot 2^{\boxed{010}_2 - 3} \cdot 1.\boxed{0000}_2$

0	0	1	0	0	0	0	0

$B = -0.03125$

$= -1^{\boxed{1}} \cdot 2^{\boxed{000}_2 - 2} \cdot 0.\boxed{0010}_2$

1	0	0	0	0	0	1	0

$A + B =$

Денормализованные `floating-point`

$A = 0.5$

$$= -1^{\boxed{0}} \cdot 2^{\boxed{010}_2 - 3} \cdot 1.\boxed{0000}_2$$

0	0	1	0	0	0	0	0

$B = -0.03125$

$$= -1^{\boxed{1}} \cdot 2^{\boxed{000}_2 - 2} \cdot 0.\boxed{0010}_2$$

1	0	0	0	0	0	1	0

$A + B =$ $2^{10_2 - 3} \cdot \left(1.0000_2 - 0.0010_2 \cdot 2^{-1}\right)$

Денормализованные `floating-point`

$\mathrm{A} = 0.5$

$$= -1^{\boxed{0}} \cdot 2^{\boxed{010}_2 - 3} \cdot 1.\boxed{0000}_2$$

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

$\mathrm{B} = -0.03125$

$$= -1^{\boxed{1}} \cdot 2^{\boxed{000}_2 - 2} \cdot 0.\boxed{0010}_2$$

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

$\mathrm{A} + \mathrm{B} =$ $2^{10_2 - 3} \cdot \left(1.0000_2 - 0.0010_2 \cdot 2^{-1}\right) =$

$$= 2^{10_2 - 3} \cdot \left(1.0000_2 - 0.00010_2\right)$$

Денормализованные `floating-point`

$A = 0.5$

$$= -1^{\boxed{0}} \cdot 2^{\boxed{010}_2 - 3} \cdot 1.\boxed{0000}_2$$

0	0	1	0	0	0	0	0

$B = -0.03125$

$$= -1^{\boxed{1}} \cdot 2^{\boxed{000}_2 - 2} \cdot 0.\boxed{0010}_2$$

1	0	0	0	0	0	1	0

$A + B =$ $2^{10_2 - 3} \cdot \left(1.0000_2 - 0.0010_2 \cdot 2^{-1}\right) =$

$$= 2^{10_2 - 3} \cdot (1.0000_2 - 0.00010_2) \approx 2^{10_2 - 3} \cdot 0.1111_2$$

Денормализованные `floating-point`

$A = 0.5$

$= -1^{\boxed{0}} \cdot 2^{\boxed{010}_2 - 3} \cdot 1.\boxed{0000}_2$

0	0	1	0	0	0	0	0

$B = -0.03125$

$= -1^{\boxed{1}} \cdot 2^{\boxed{000}_2 - 2} \cdot 0.\boxed{0010}_2$

1	0	0	0	0	0	1	0

$A + B =$ $2^{10_2 - 3} \cdot \left(1.0000_2 - 0.0010_2 \cdot 2^{-1}\right) =$

$= 2^{10_2 - 3} \cdot (1.0000_2 - 0.00010_2) \approx 2^{10_2 - 3} \cdot 0.1111_2 =$

$= -1^{\boxed{0}} \cdot 2^{\boxed{001}_2 - 3} \cdot 1.\boxed{1110}_2$ $= 0.46875$

0	0	0	1	1	1	1	0

Умножение `floating-point`

$\mathrm{A} = 3.125 \approx \pi$

$= -1^{\boxed{0}} \cdot 2^{\boxed{100}_2 - 3} \cdot 1.\boxed{1001}_2$

0	1	0	0	1	0	0	1

$\mathrm{B} = 2.625 \approx e$

$= -1^{\boxed{0}} \cdot 2^{\boxed{100}_2 - 3} \cdot 1.\boxed{0101}_2$

0	1	0	0	0	1	0	1

$\mathrm{A} \cdot \mathrm{B} = -1^{0+0} \cdot 2^{100_2 - 3 + 100_2 - 3} \cdot 1.1001_2 \cdot 1.0101_2$

Умножение `floating-point`

$\mathrm{A} = 3.125 \approx \pi$

$= -1^{\boxed{0}} \cdot 2^{\boxed{100}_2 - 3} \cdot 1.\boxed{1001}_2$

0	1	0	0	1	0	0	1

$\mathrm{B} = 2.625 \approx e$

$= -1^{\boxed{0}} \cdot 2^{\boxed{100}_2 - 3} \cdot 1.\boxed{0101}_2$

0	1	0	0	0	1	0	1

$\mathrm{A} \cdot \mathrm{B} = -1^{0+0} \cdot 2^{100_2 - 3 + 100_2 - 3} \cdot 1.1001_2 \cdot 1.0101_2 =$

$= -1^{0} \cdot 2^{101_2 - 3} \cdot 10.00001101_2$

Умножение `floating-point`

$A = 3.125 \approx \pi$

$= -1^{\boxed{0}} \cdot 2^{\boxed{100}_2 - 3} \cdot 1.\boxed{1001}_2$

0	1	0	0	1	0	0	1

$B = 2.625 \approx e$

$= -1^{\boxed{0}} \cdot 2^{\boxed{100}_2 - 3} \cdot 1.\boxed{0101}_2$

0	1	0	0	0	1	0	1

$A \cdot B = -1^{0+0} \cdot 2^{100_2 - 3 + 100_2 - 3} \cdot 1.1001_2 \cdot 1.0101_2 =$

$= -1^{0} \cdot 2^{101_2 - 3} \cdot 10.00001101_2 \approx$

$\approx -1^{\boxed{0}} \cdot 2^{\boxed{110}_2 - 3} \cdot 1.\boxed{0000}_2 = 8$

0	1	1	0	0	0	0	0

Спасибо за внимание!

🔗 github.com/JakMobius/courses/tree/main/mipt-os-basic-2024