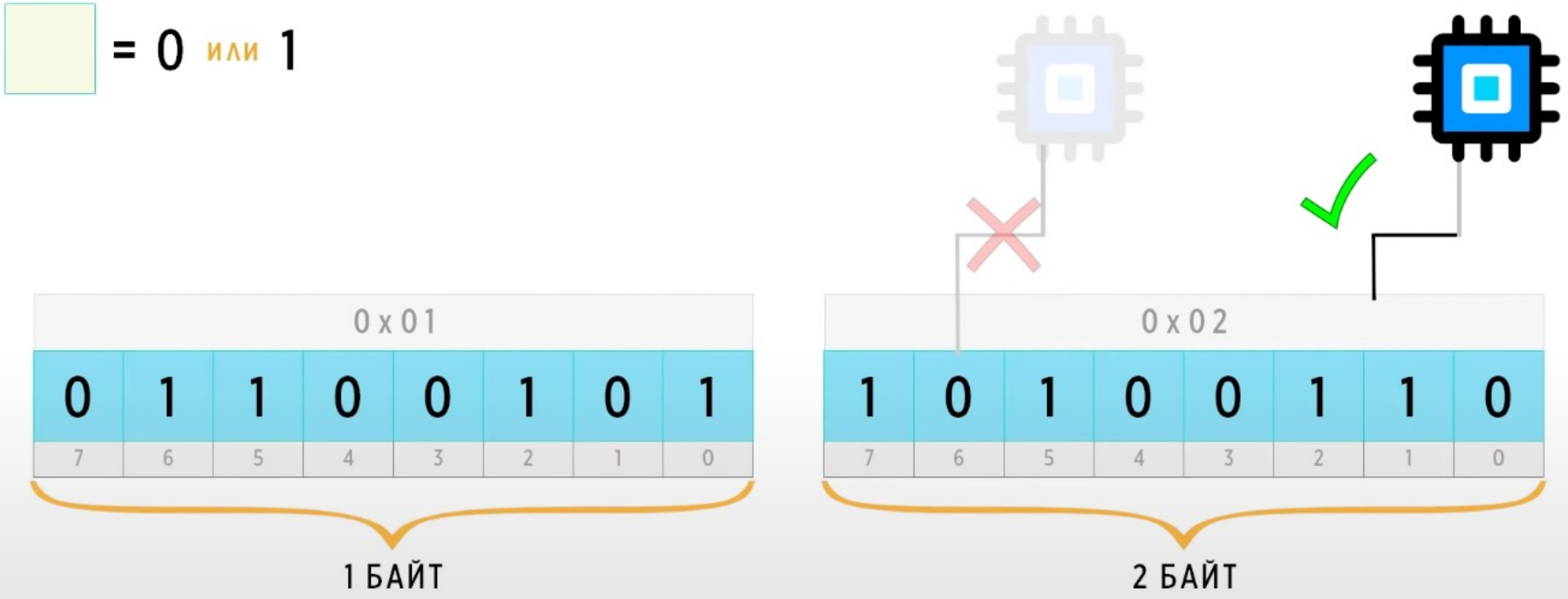
НИЗКОУРОВНЕВОЕ ПРОГРАММИРОВАНИЕ

ЛЕКЦИЯ 3

Оперативная память, кодировки, стек

Абстрактная модель оперативной памяти

- **Бит** - минимальная единица хранения информации



Абстрактная модель оперативной памяти

Байт = ▢▢▢▢▢▢▢▢…

0x01	0x02	0x03	0x04	0x05
h	e	l	l	o

Символ	Десятичный код	Двоичный код
A	65	010000001
B	66	01000010
C	67	01000011
D	68	01000100
E	69	01000101
F	70	01000110
G	71	01000111
H	72	01001000
I	73	01001001
J	74	01001010
K	75	01001011
L	76	01001100
M	77	01001101

ASCII

0	1	1	0	0	1	0	1

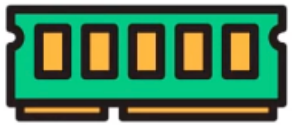
от 0 до 127

0	1	1	0	0	0	0	1

a == 97 == 1100001

Размер оперативной памяти

2048 == 2гб 4096 == 4гб 8192 == 8гб



Регистр процессора

1	1	1	1	1	1	1	1

Количество адресов ОЗУ

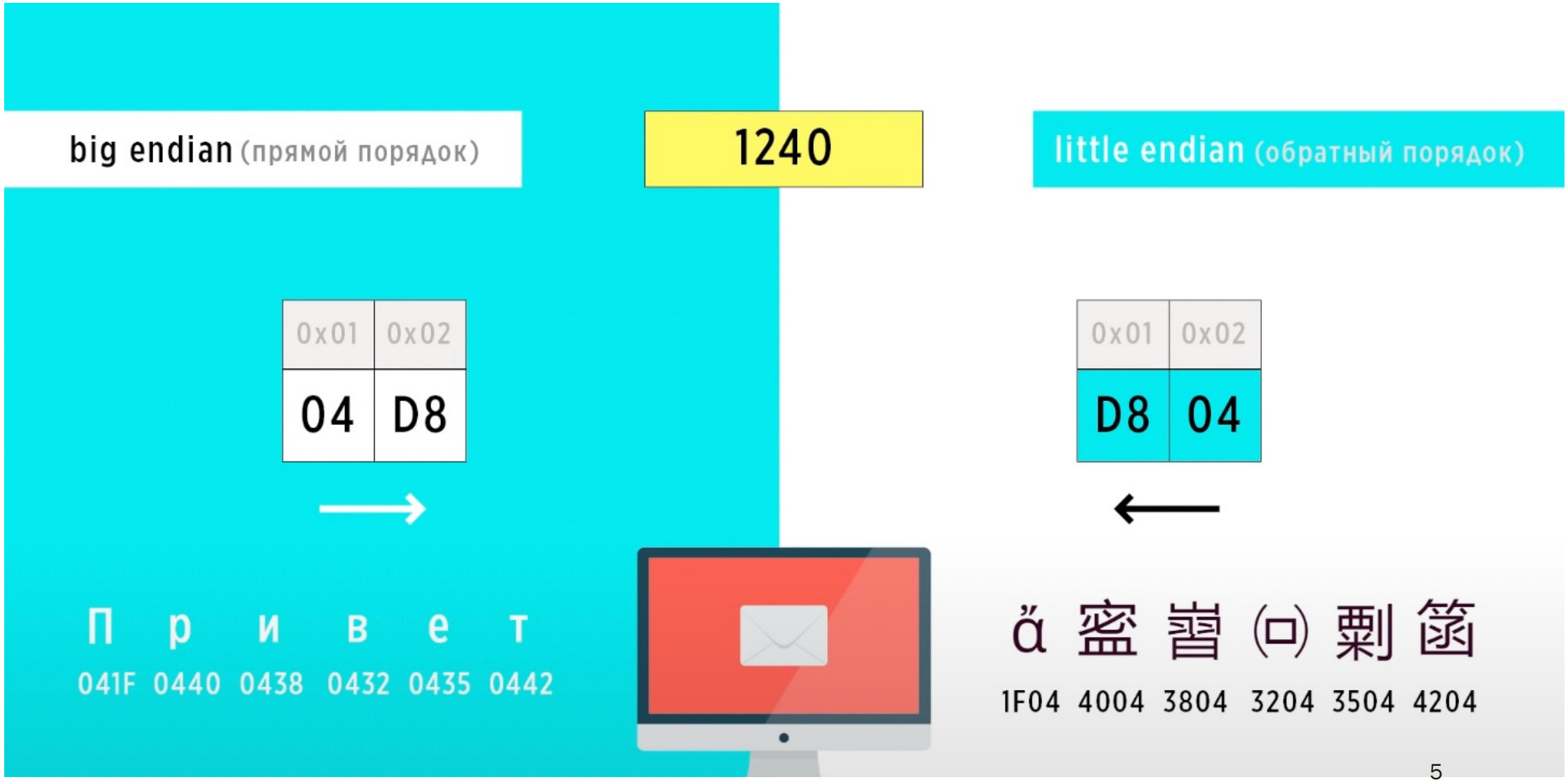
Максимальное число = $2^{\text{размер регистра}}$

Регистр на 8 бит = 2^8 = 256 байт

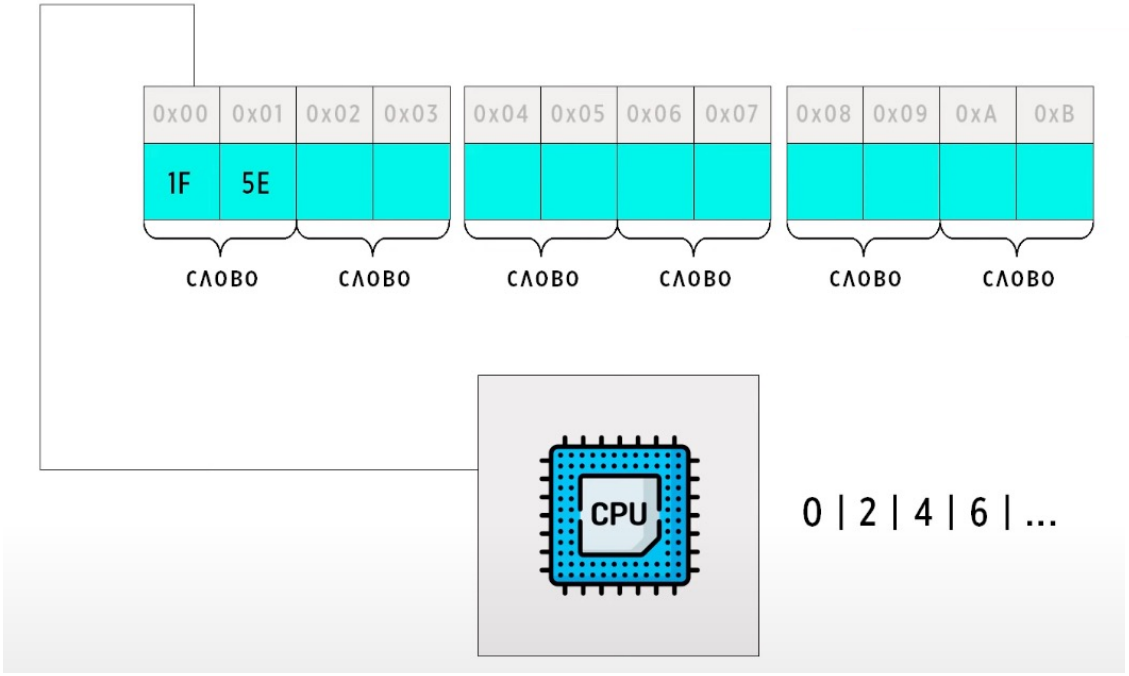
Регистр на 10 бит = 2^{10} = 1024 байта

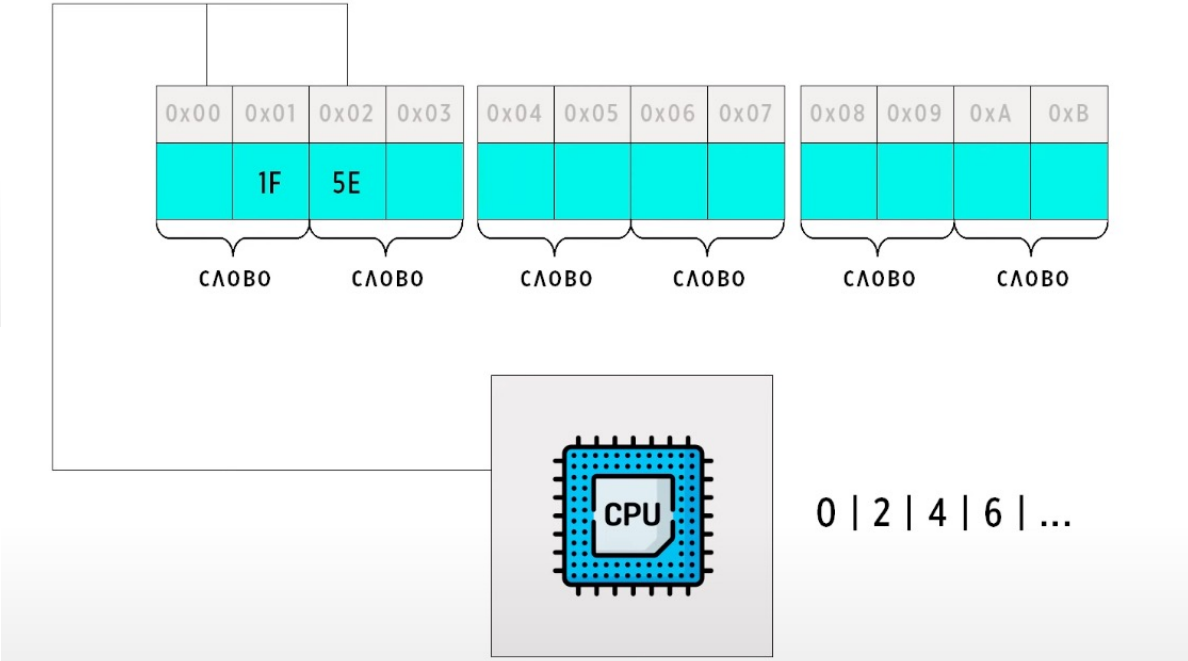
Регистр на 32 бита = 2^{32} = 4294967296 байт = 4096мб = 4гб

Хранение данных в памяти

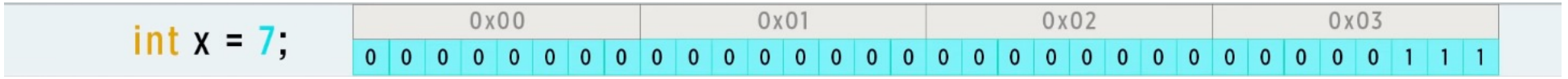


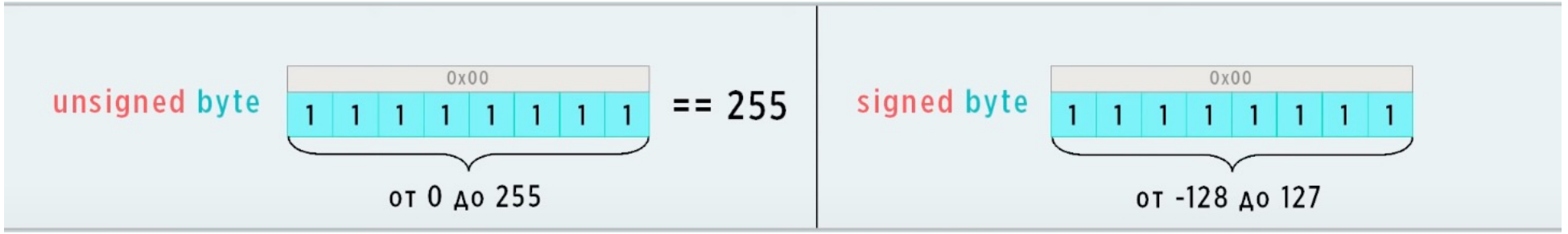
Выравнивание данных

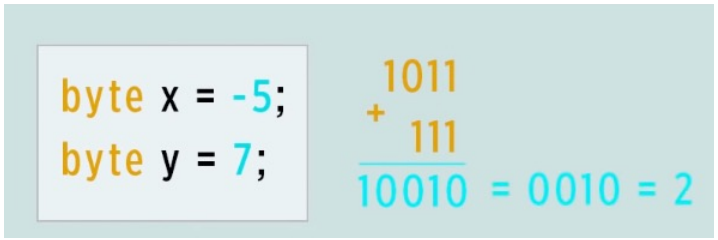


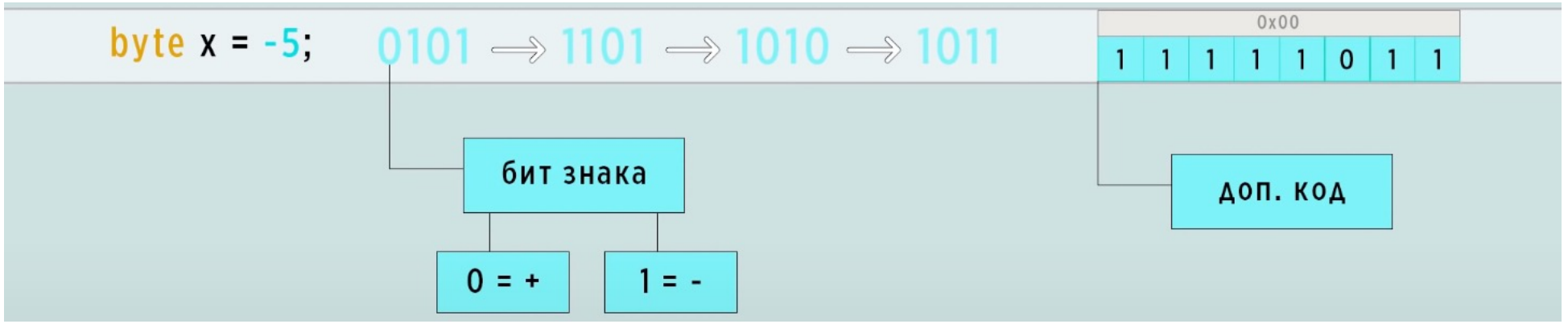


Числовые типы данных









Ссылочные типы данных

Массив это структура данных, в которой каждый элемент имеет индекс от 0 до n-1.

```
byte arr[] = {2, 4, 5, 7, 9};
```

1. Адрес начала массива
2. Индекс искомого элемента
3. Размер элементов массива

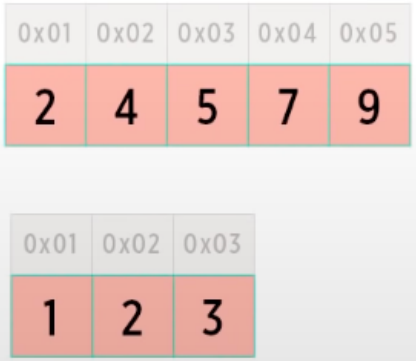
Сборщик мусора

Автоматическое управление

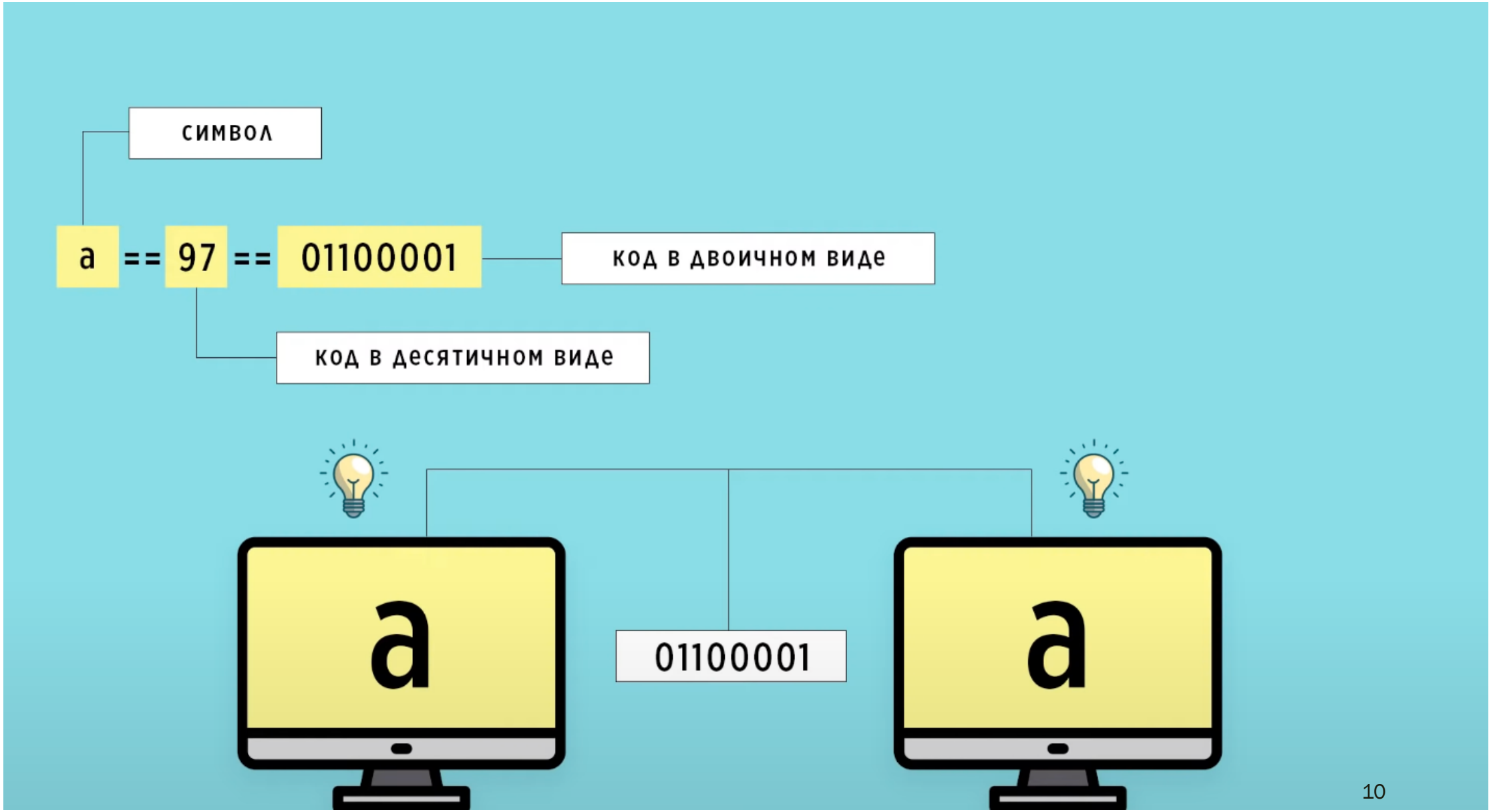
```
byte[] arr = {2, 4, 5, 7, 9};

arr = new byte[]{1, 2, 3};
```

Ручное управление

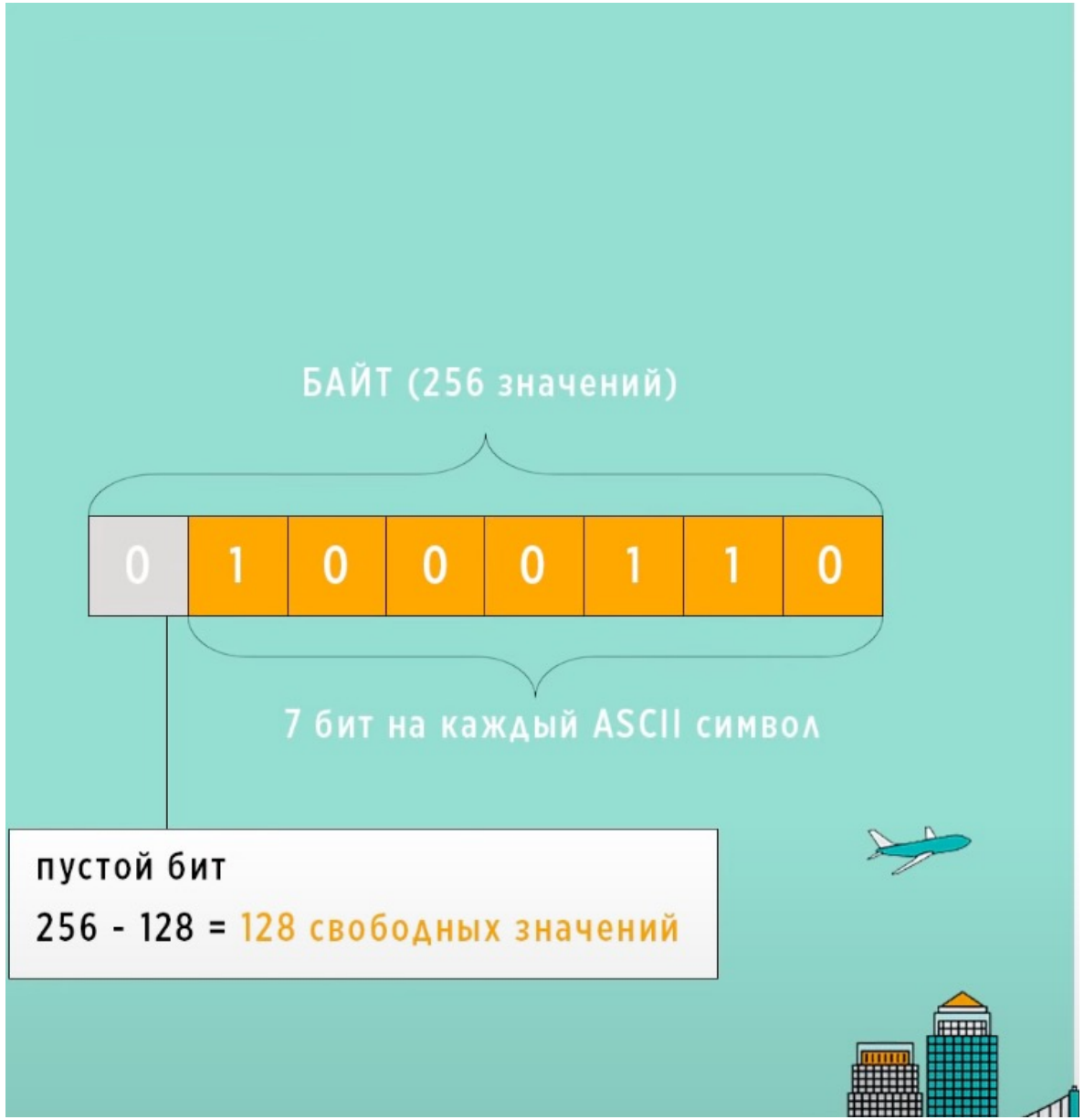
```
int *arr = new int[5]{2, 4, 5, 7, 9};
delete[] arr;
arr = new int[3]{1, 2, 3};
```



Как работают кодировки



Как работают кодировки



ASCII

American standart code for information interchange

- от 0 до 127
- 128 значений
- 31 символ управляющий (LF, CR, ...)
- с 31 до конца имеют внешний вид

Кодовая страница

code page

- набор из 256 символов
- первые 128 символов совпадают с ASCII
- остальные 128 символов для конкретного языка

Как работают кодировки

тысячи символов

8 бит = 2^8 = 256 значений

0	1	0	0	0	1	1	0

16 бит = 2^{16} = 65 536 значений

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

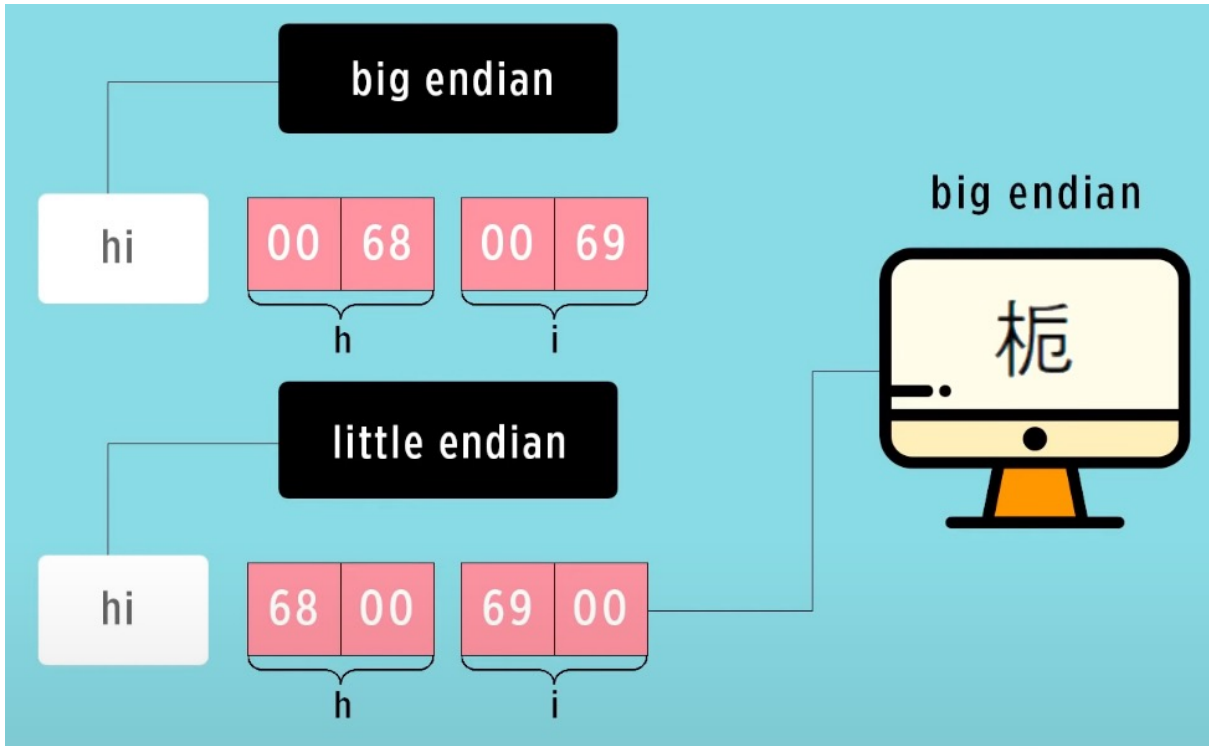
ANSI

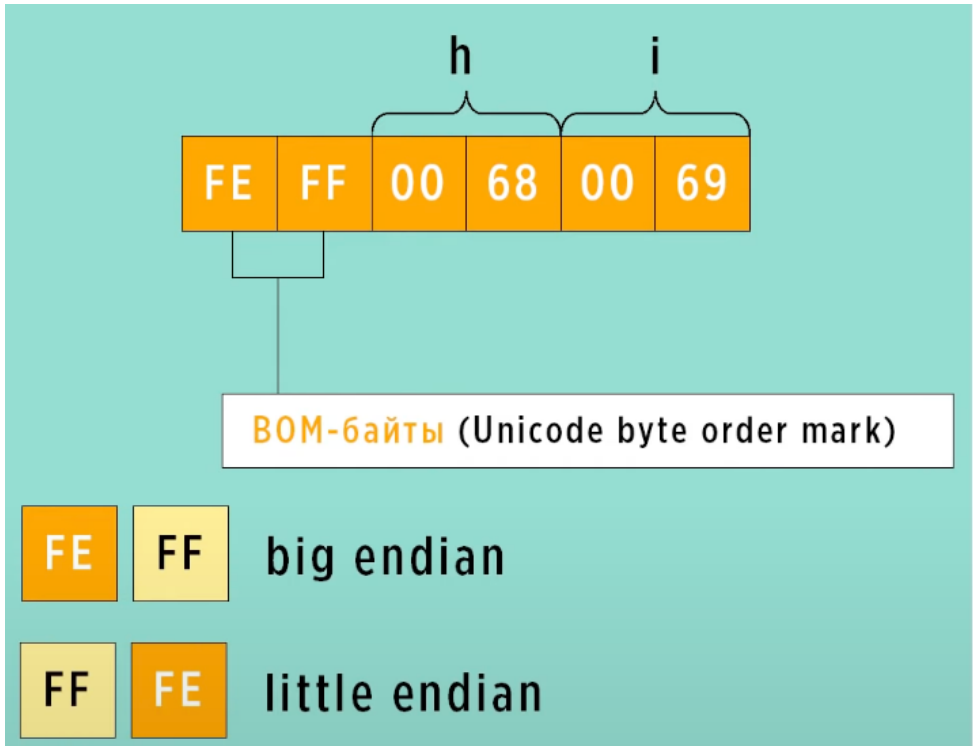
American National Standards Institute

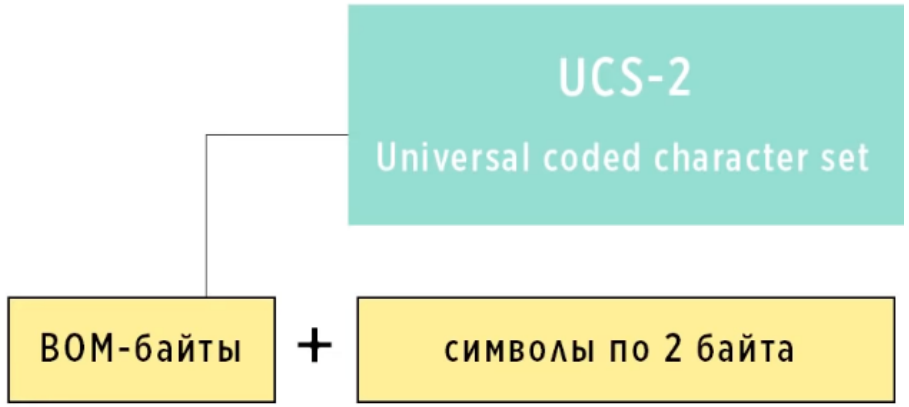
- **windows-125X** — кодовые страницы
  - **X** — номер страны
- **windows-1251** — кириллица

Как работают кодировки

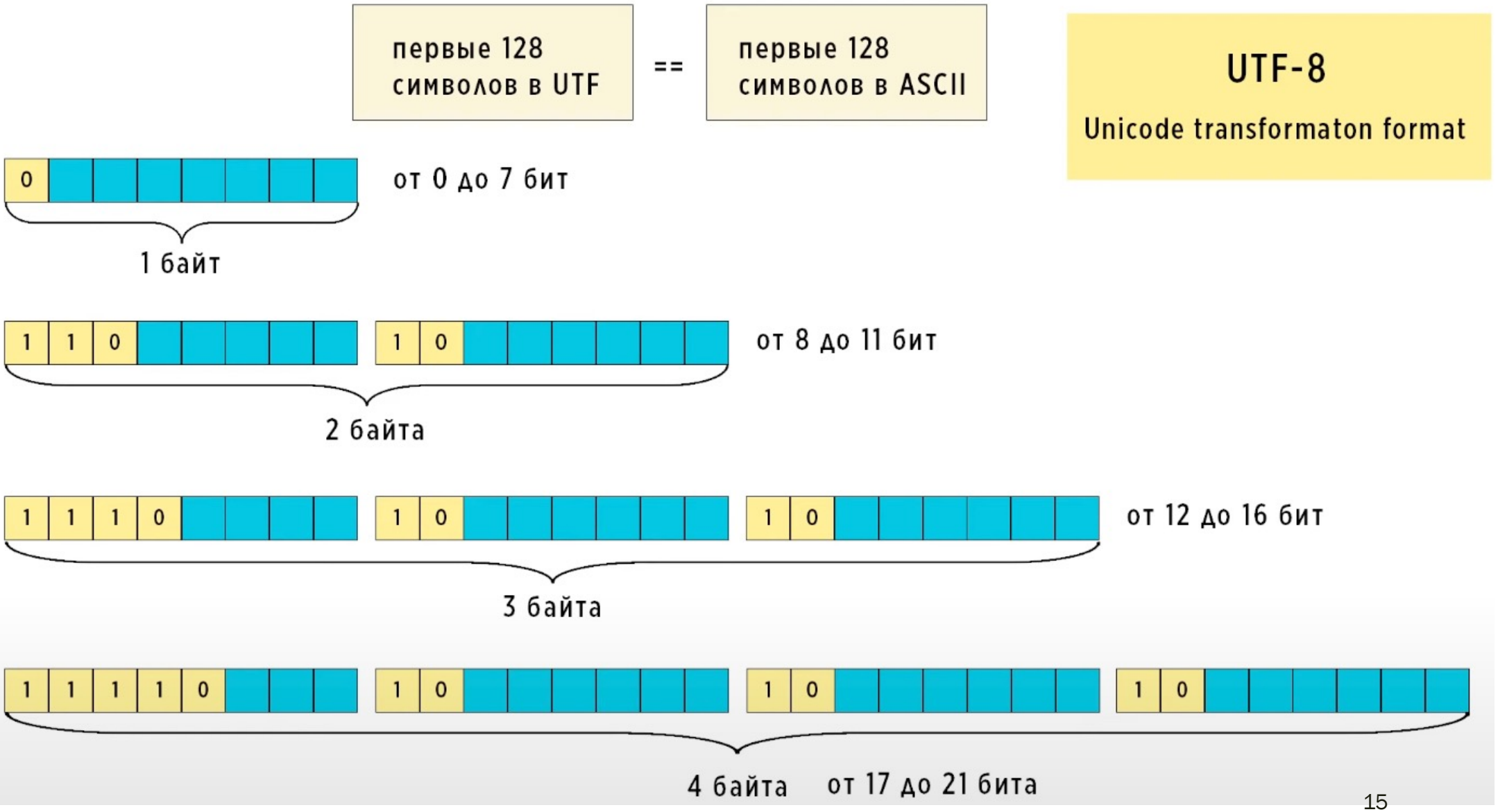
Порядок байт в Unicode







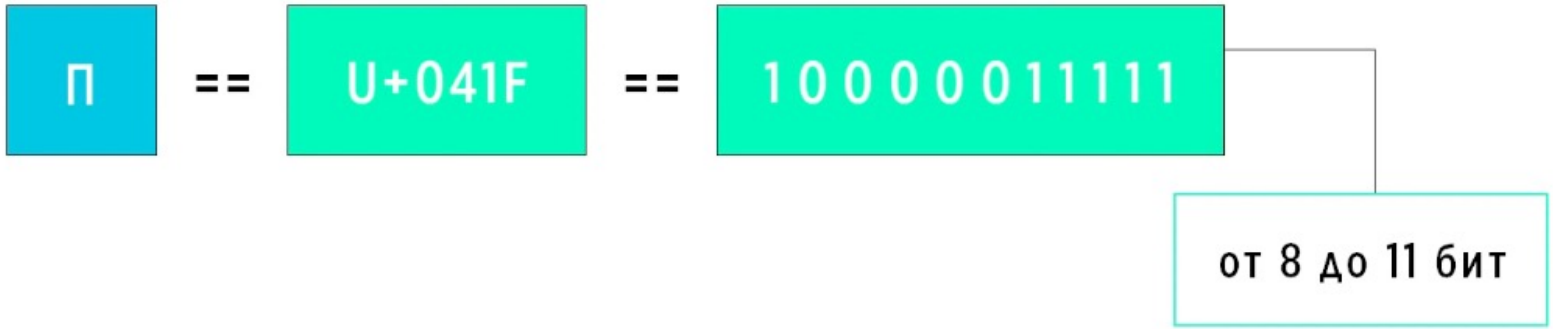
Кодировка UTF-8

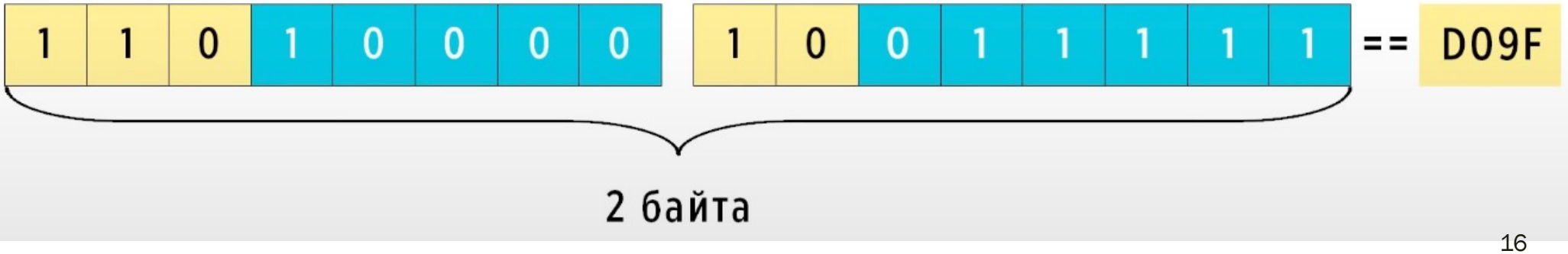


Кодировка UTF-8

UTF-8

Unicode transformaton format





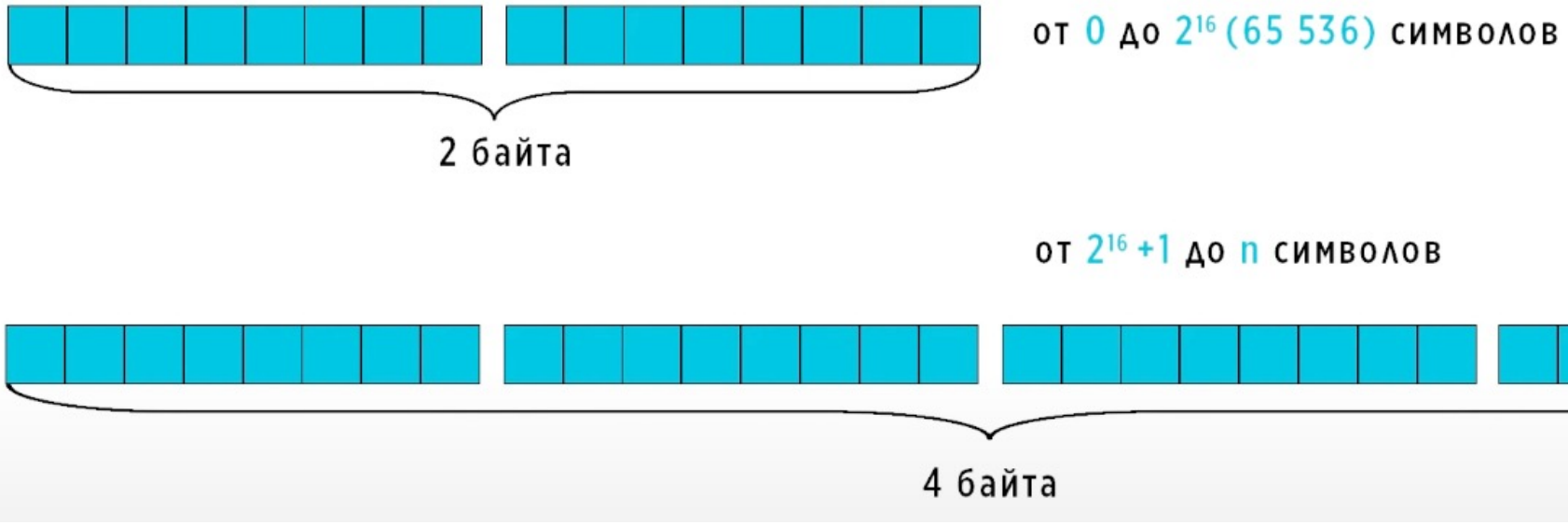
Кодировка UTF-16

UTF-16

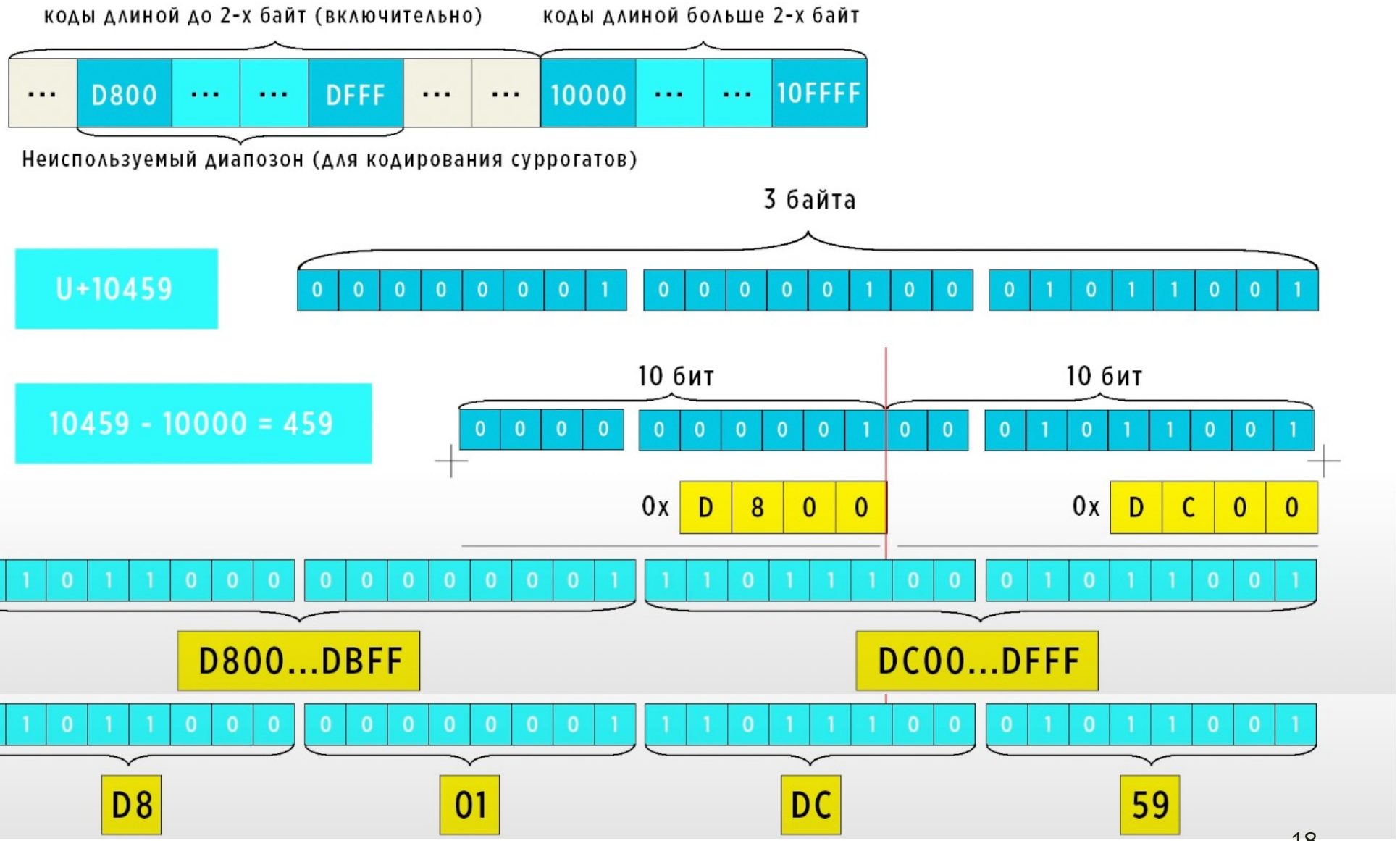
Unicode transformaton format

совместимость со старой кодировкой

хранить больше чем 2 байта

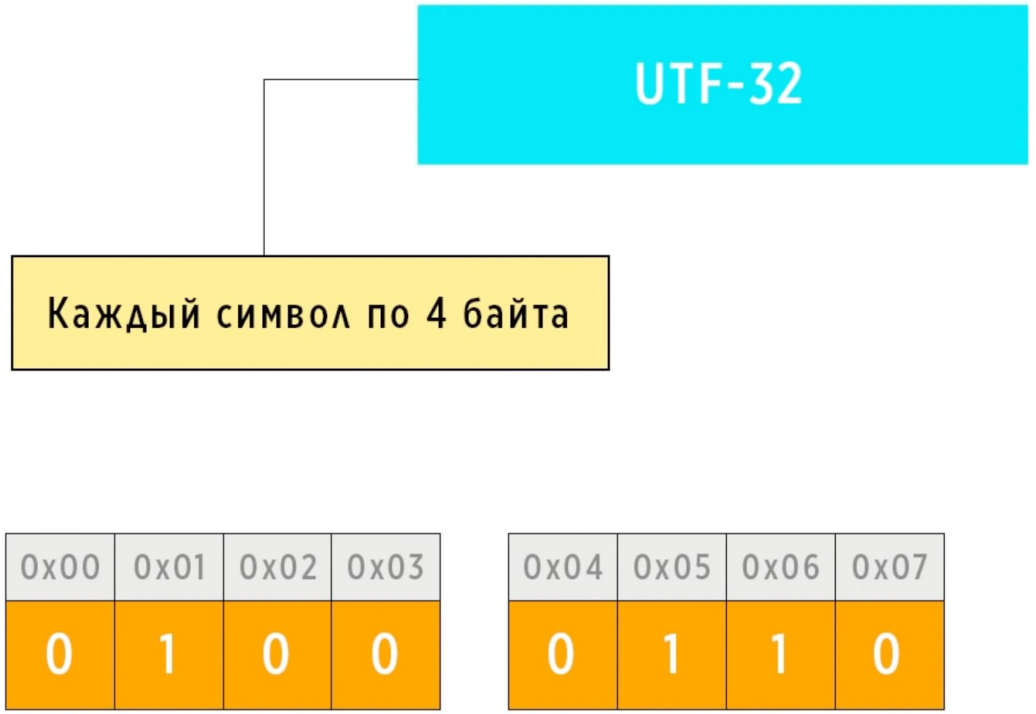


Суррогатная пара



UTF-32

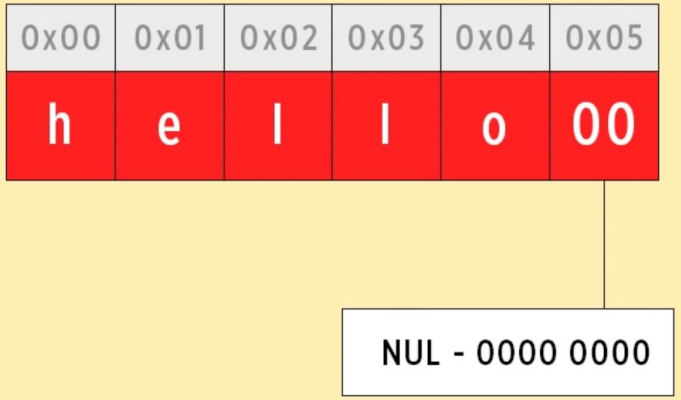
	Потребление памяти	BOM	Скорость обработки
UTF-8	от 1 до 4 байт	✗	● ●
UTF-16	2 или 4 байта	✓	● ● ●
UTF-32	4 байта	✓	● ● ● ●



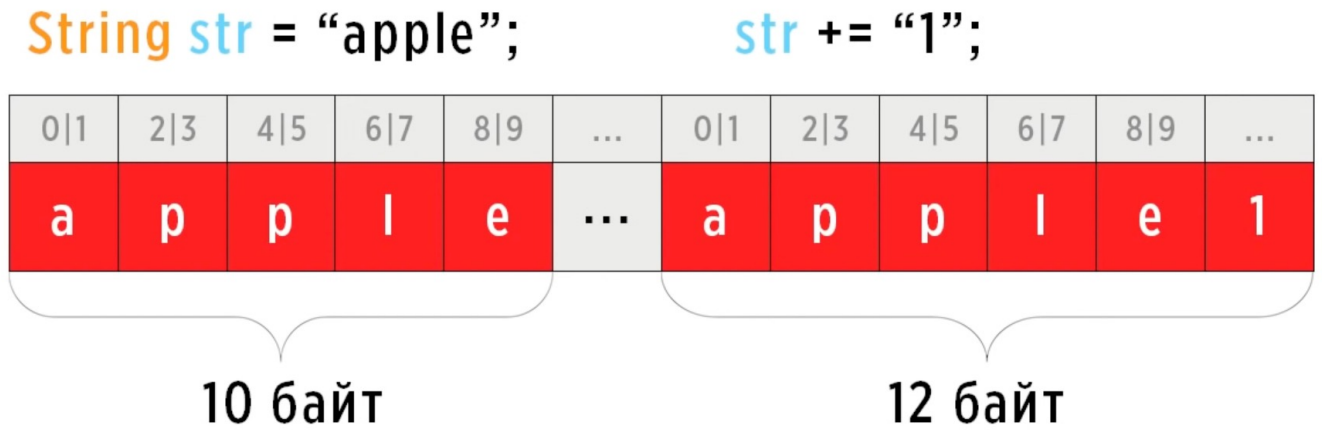
Строки в языках программирования

Строки в памяти

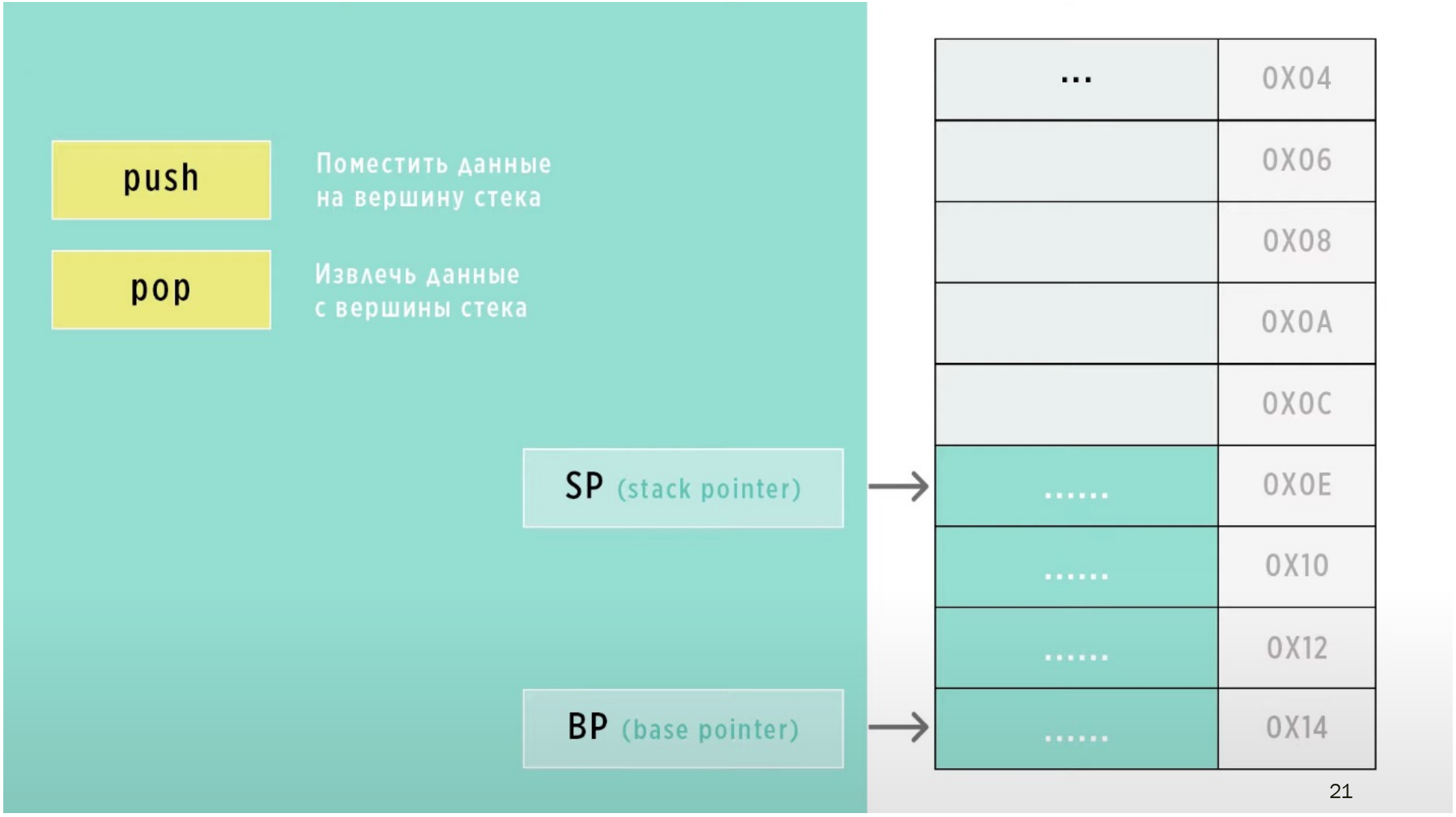
СИ-string



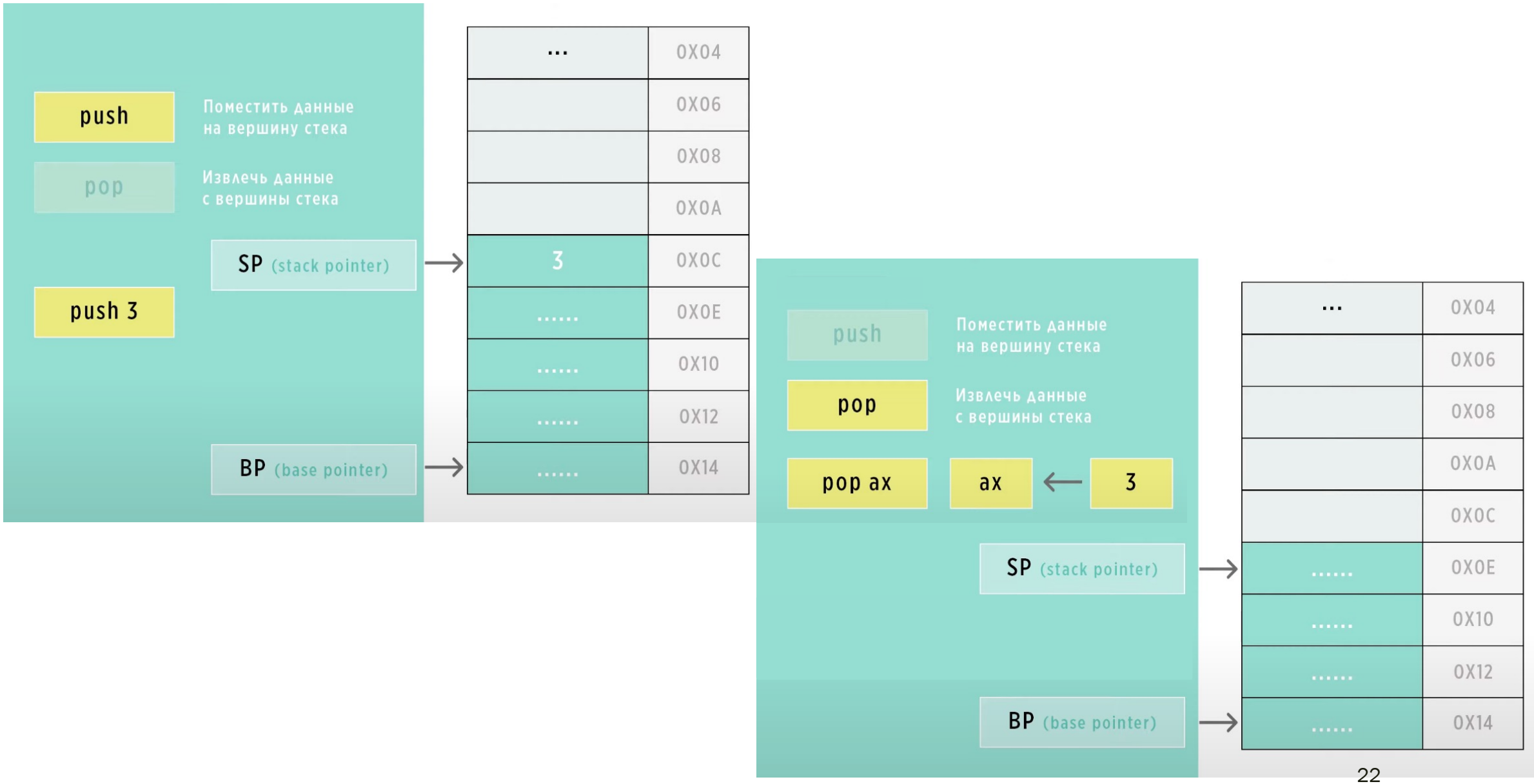
Неизменяемые строки



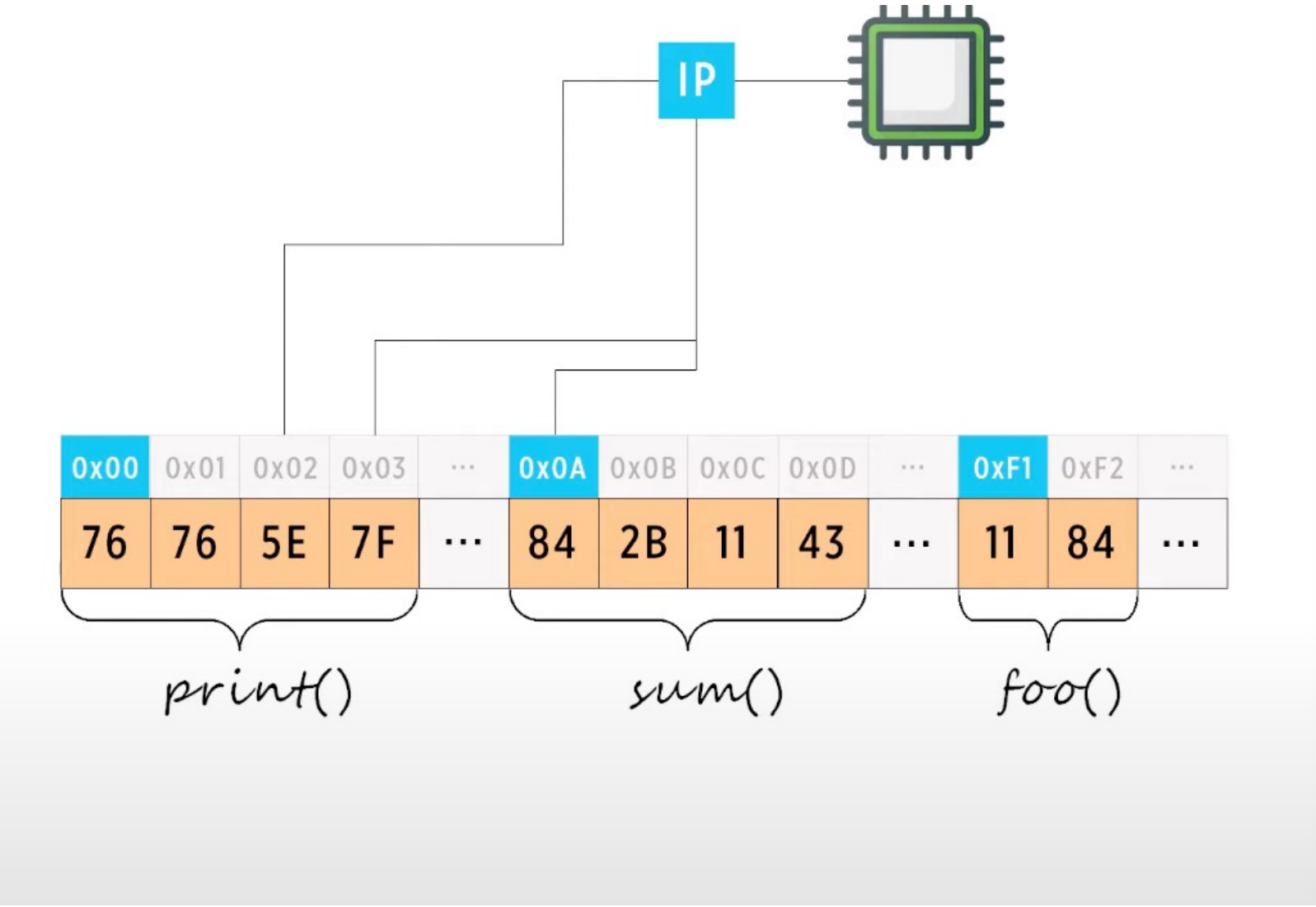
Как работает стек



Как работает стек

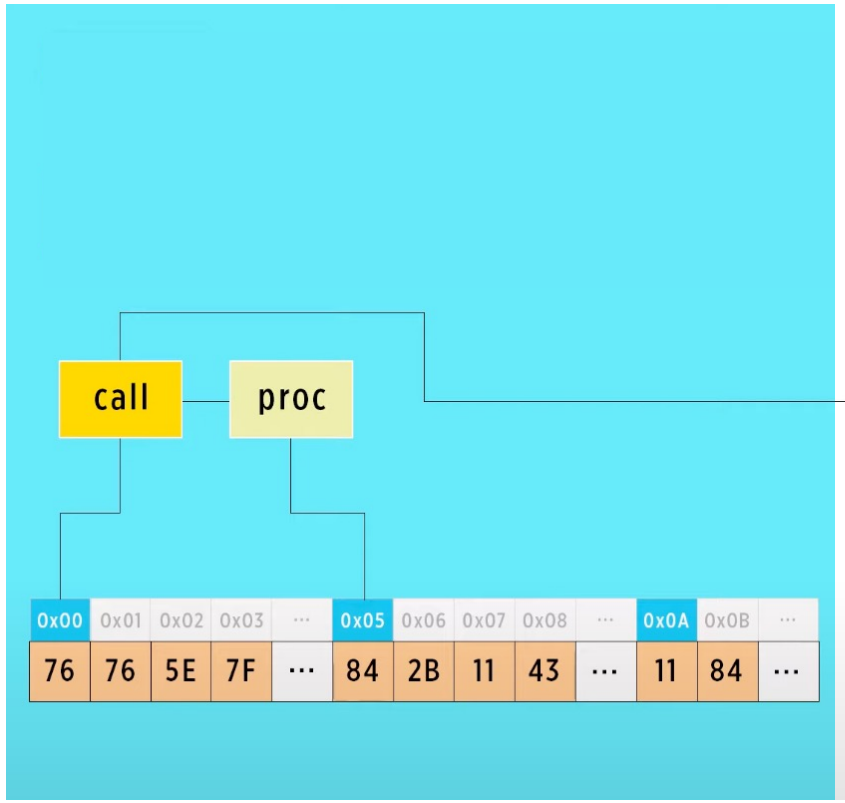


Стек и функции



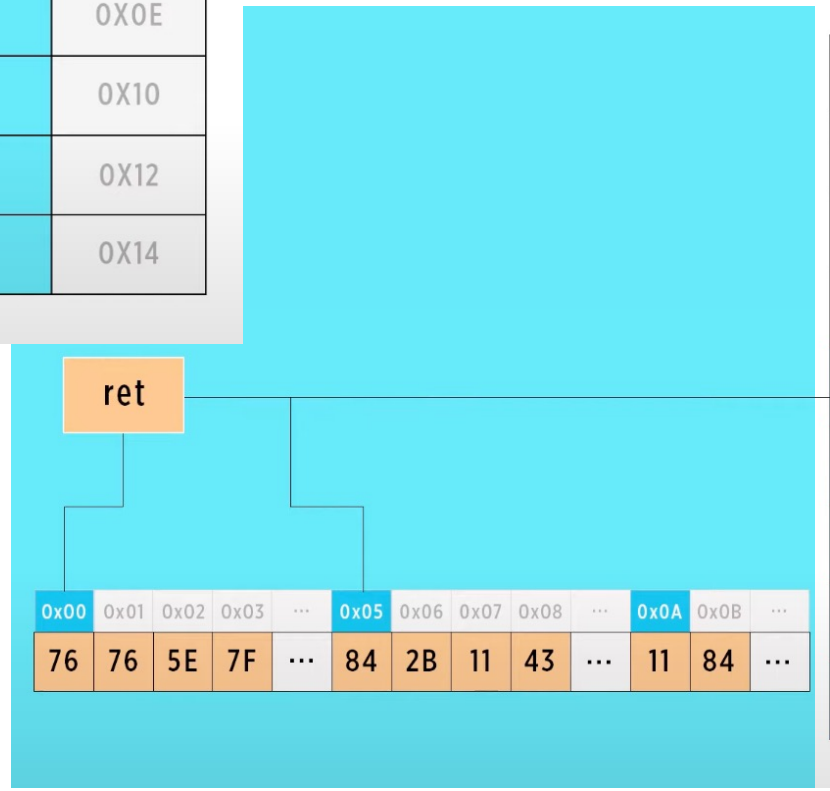

```
int sum(int x, int y) {
  return x + y;
}

void print() {
  cout << sum(1, 2);
}

void foo() {
  // ..........
}
```

Переход между подпрограммами



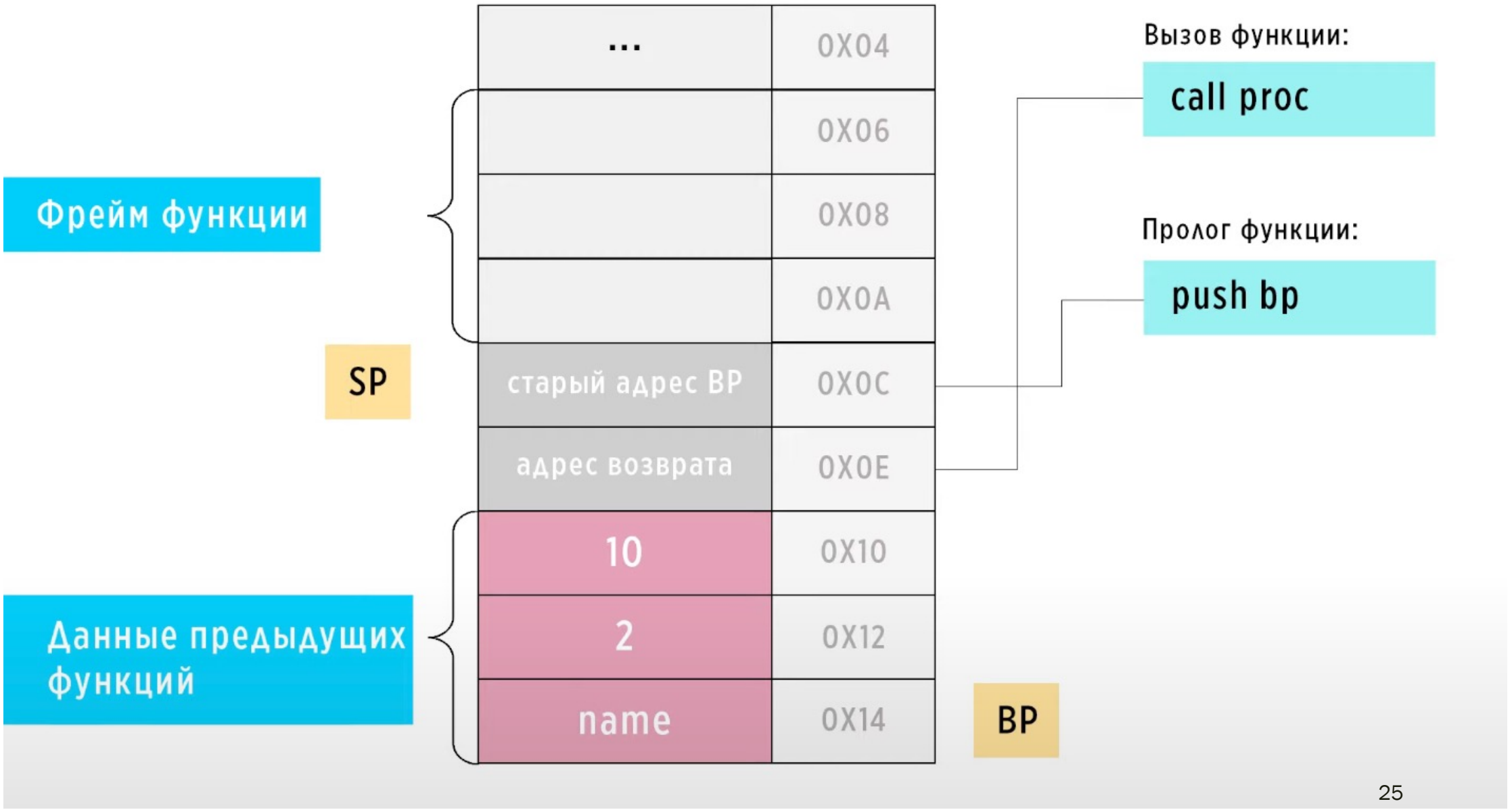
···	0X04
	0X06
1	0X08
1	0X0A
адрес возврата	0X0C
.....	0X0E
.....	0X10
.....	0X12
.....	0X14

jmp call ret

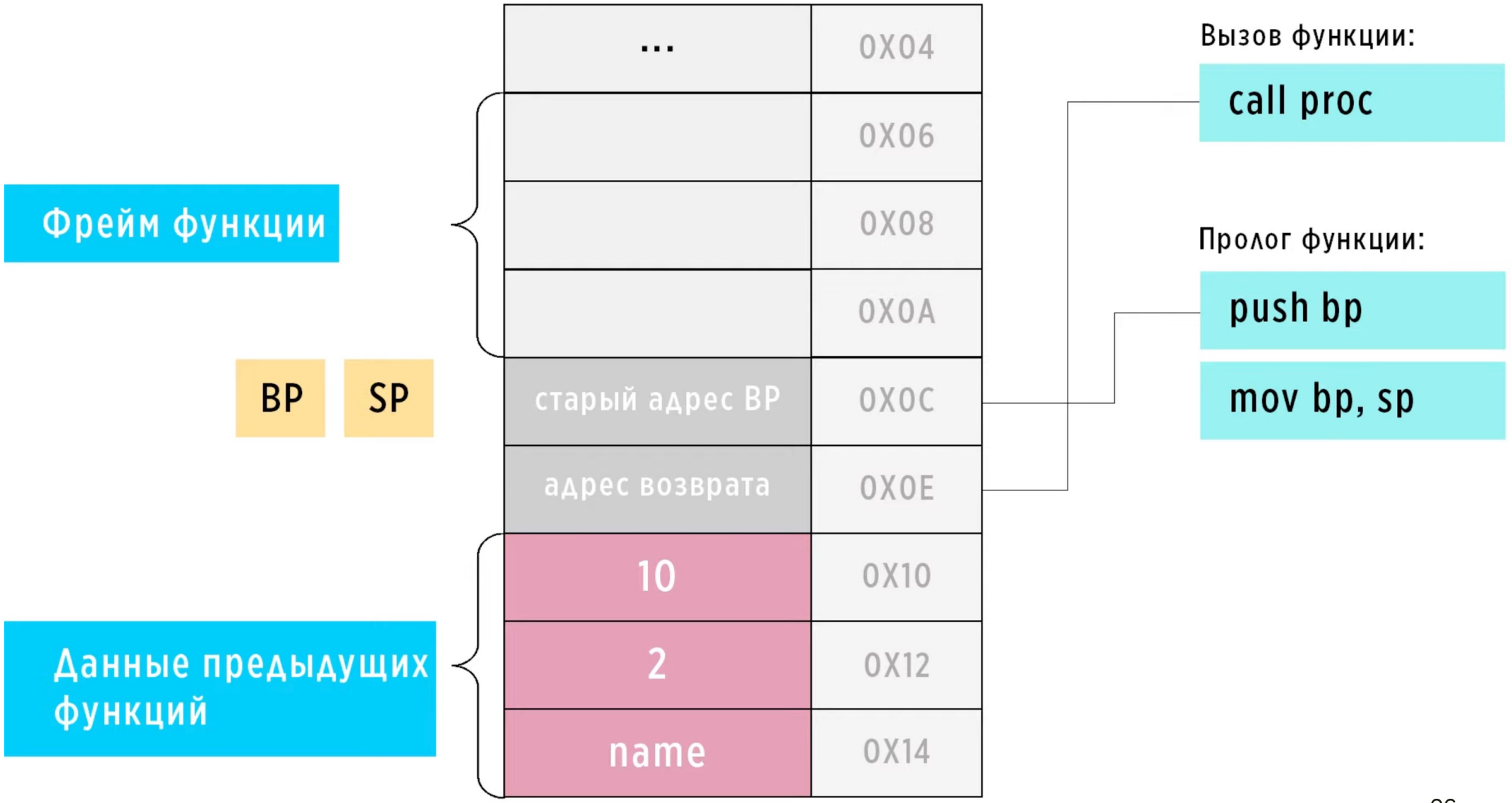


···	0X04
	0X06
1	0X08
1	0X0A
адрес возврата	0X0C
.....	0X0E
.....	0X10
.....	0X12
.....	0X14

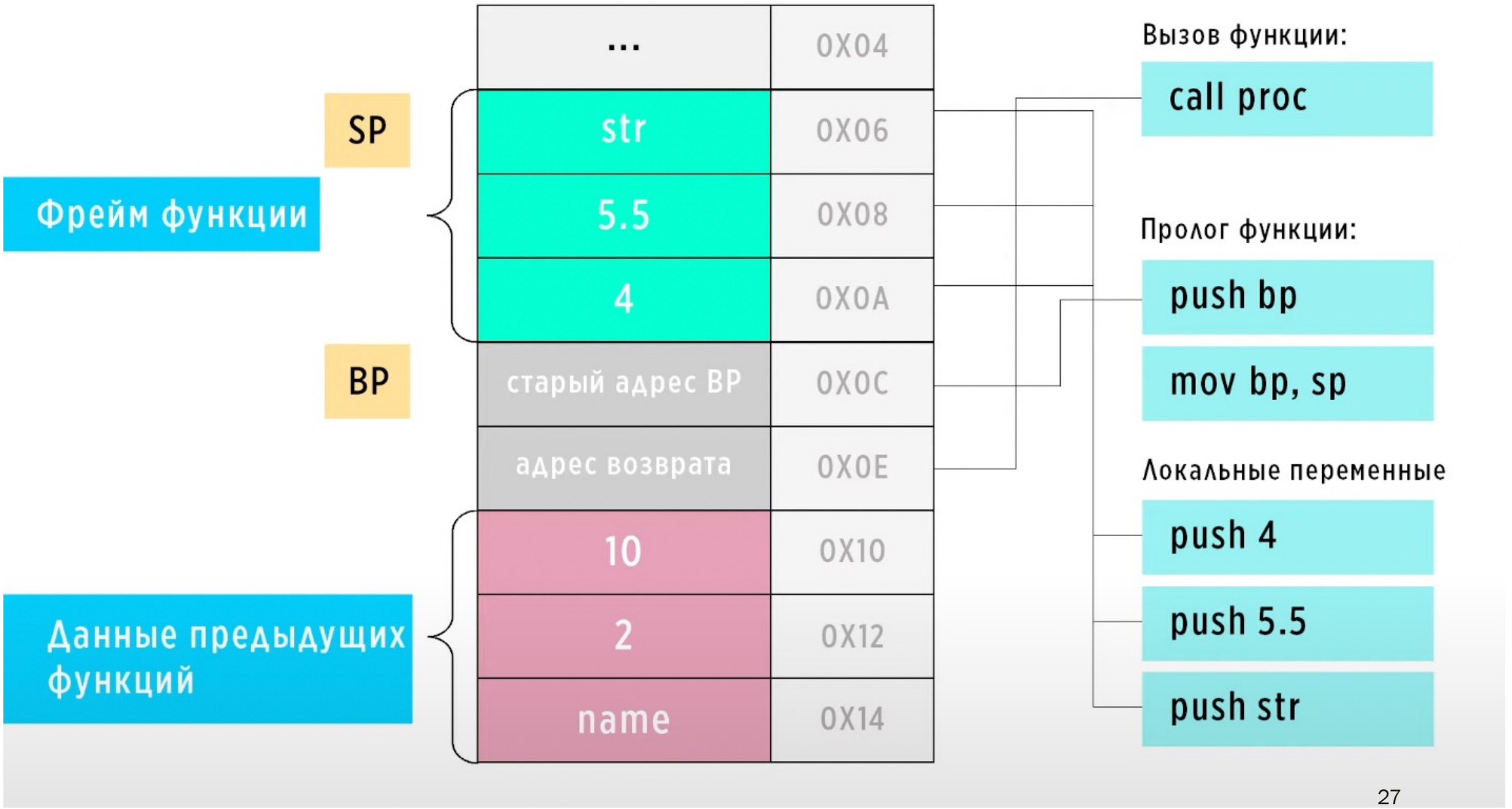
Вызов функции



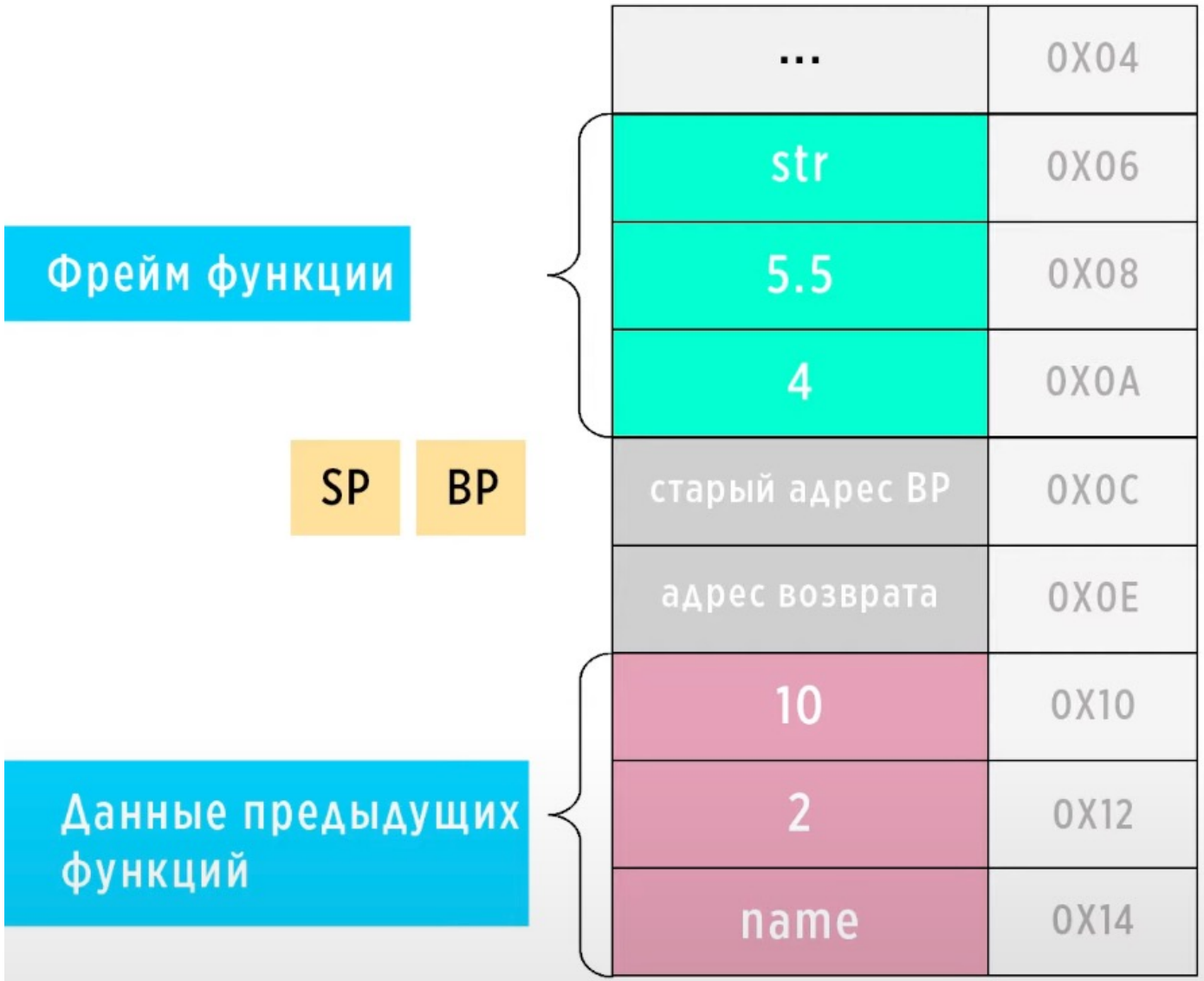
Вызов функции



Вызов функции



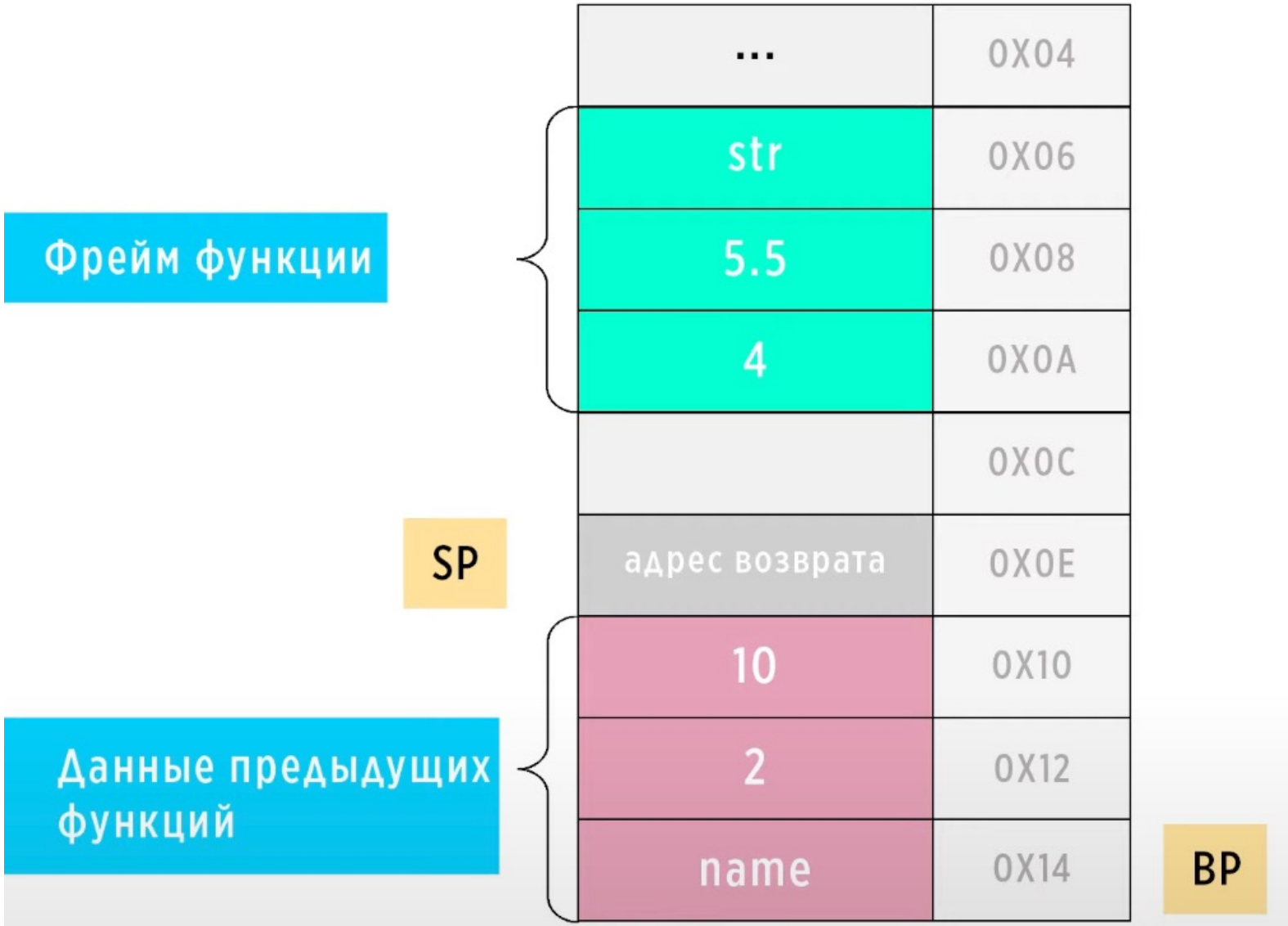
Вызов функции



Эпилог функции:

```
mov sp, bp
```

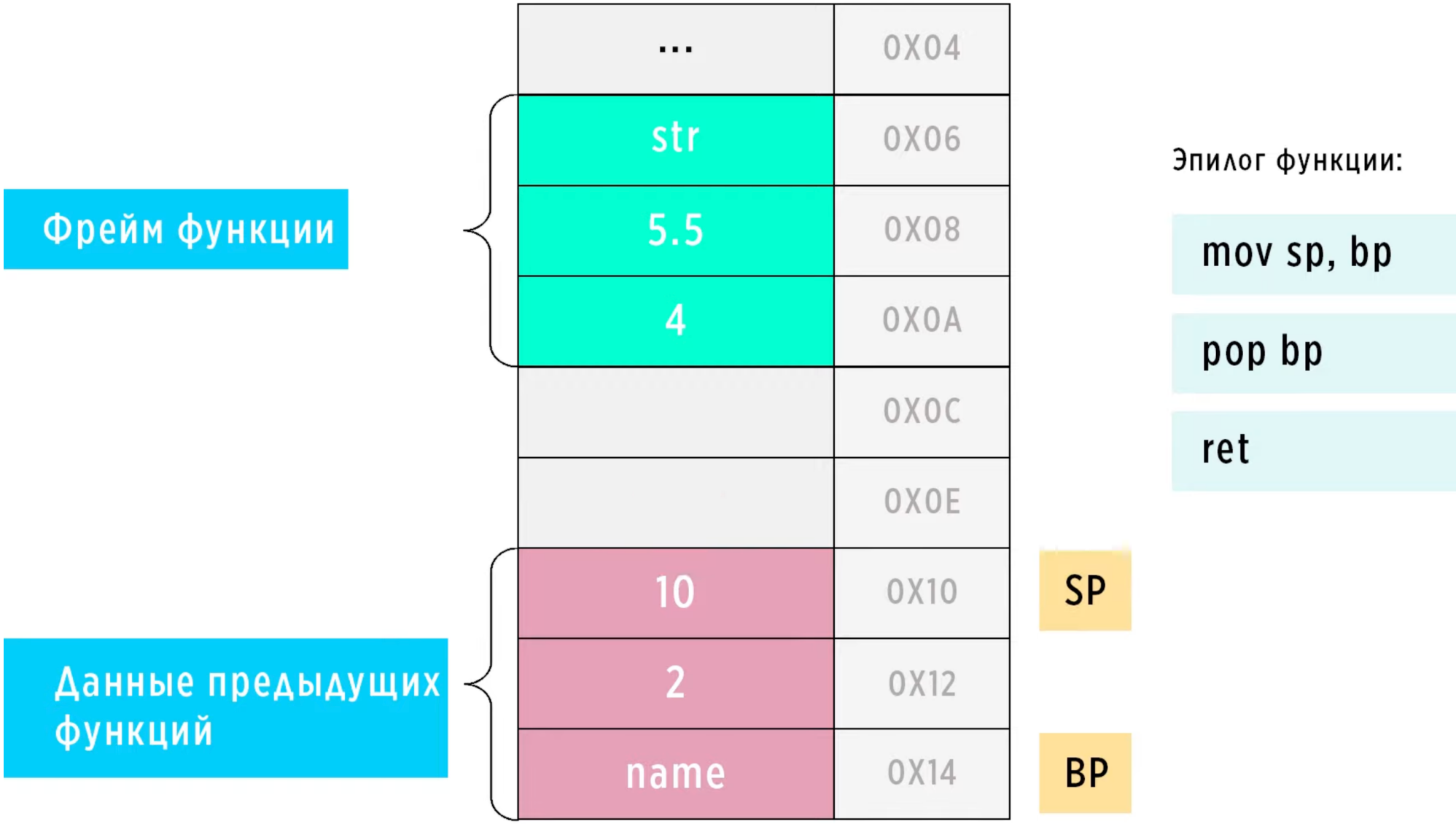
Вызов функции



Эпилог функции:

```
mov sp, bp
pop bp
```

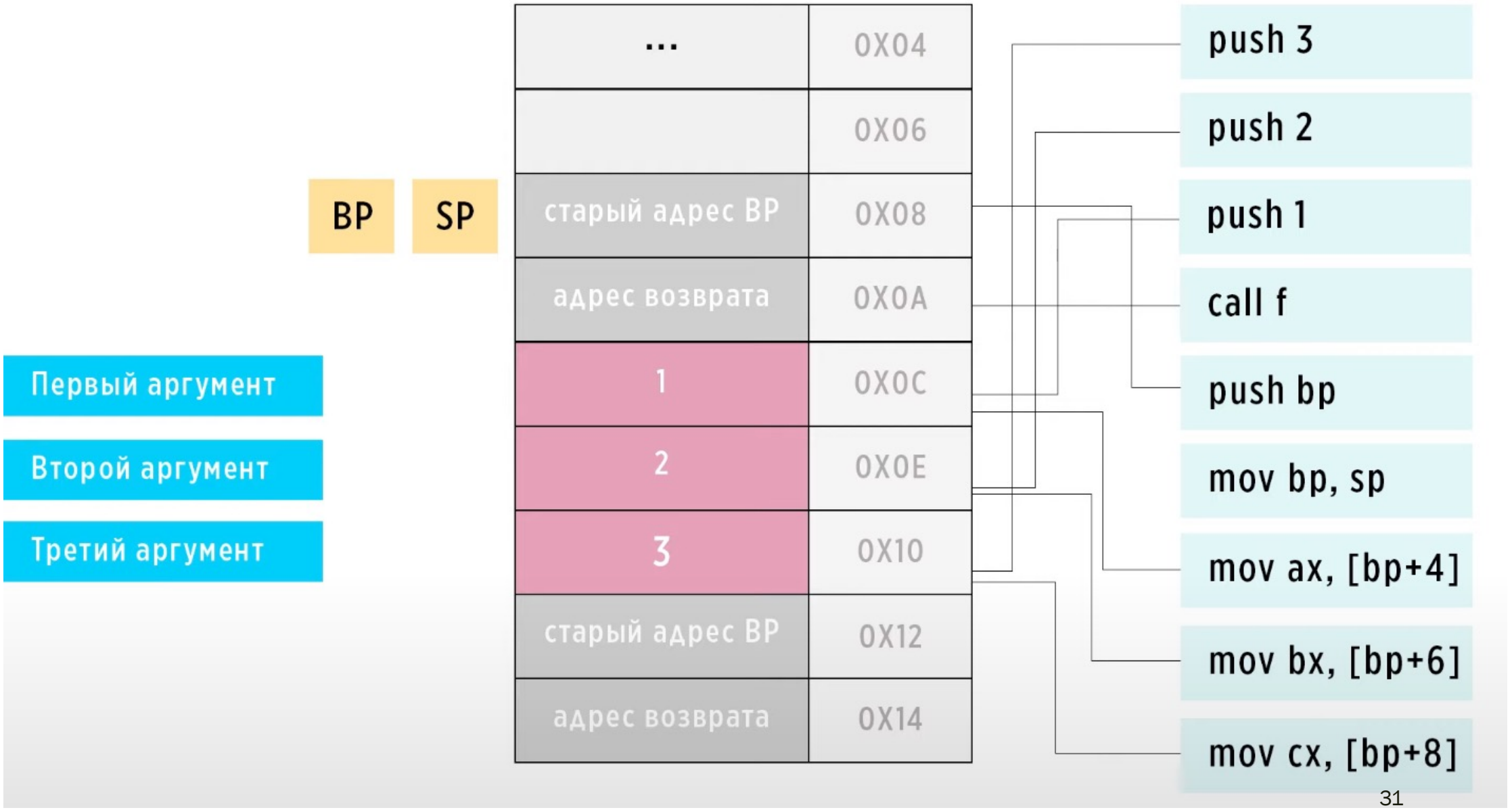
Как работает стек



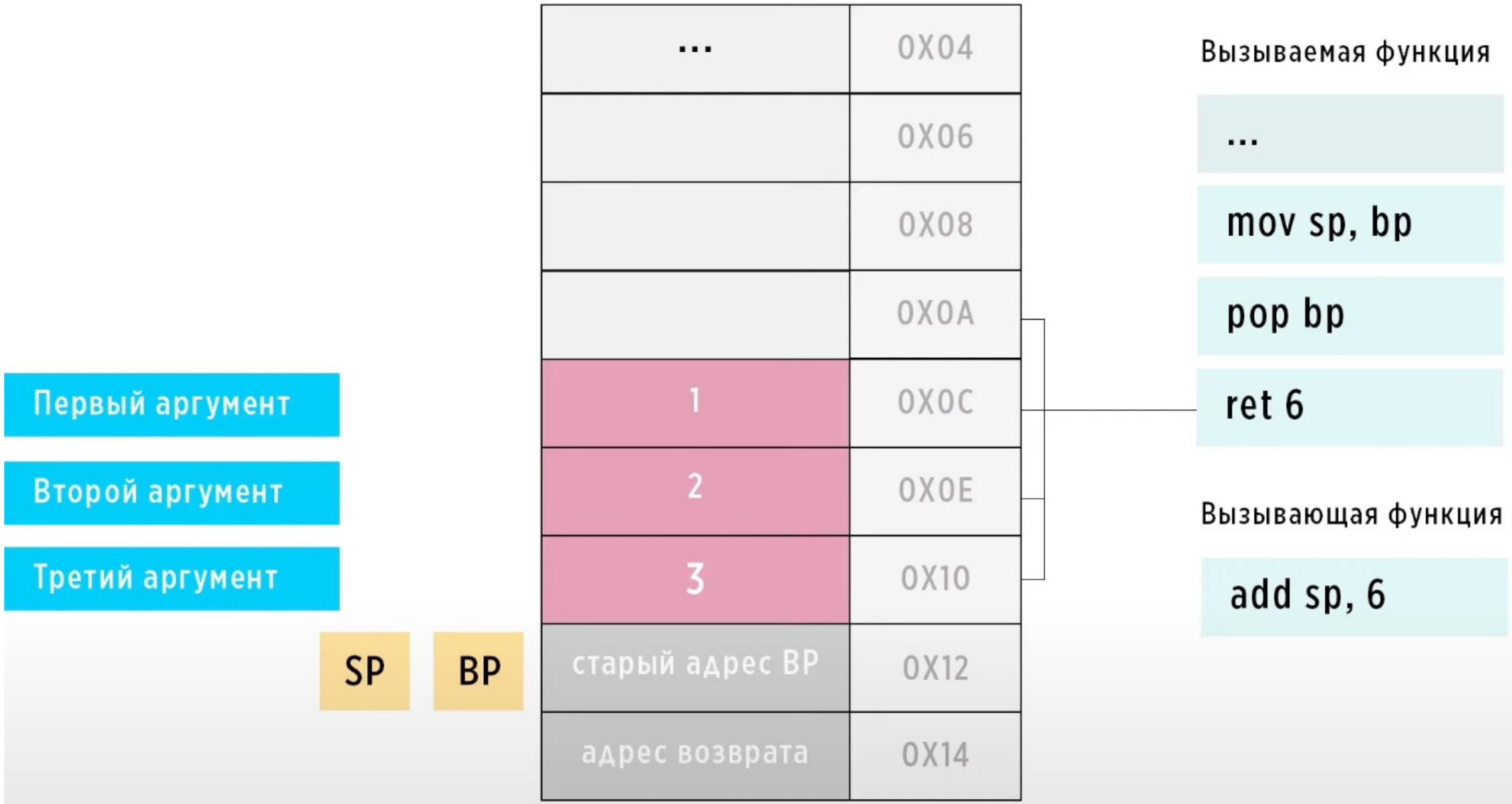
Эпилог функции:

```
mov sp, bp
pop bp
ret
```

Передача параметров в функцию

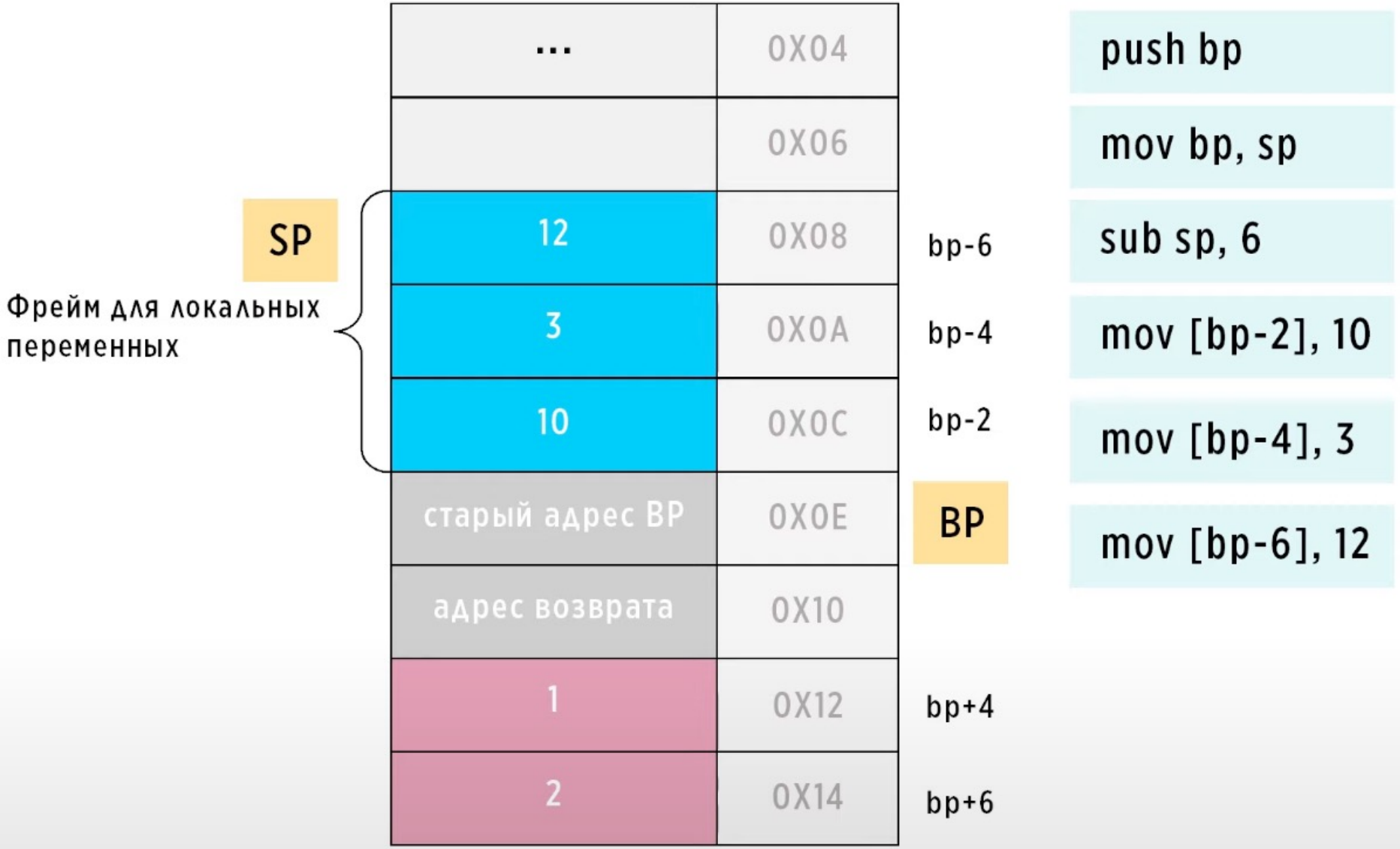


Как работает стек



Способы передачи аргументов в функцию

Локальные переменные функции



Локальные переменные функции