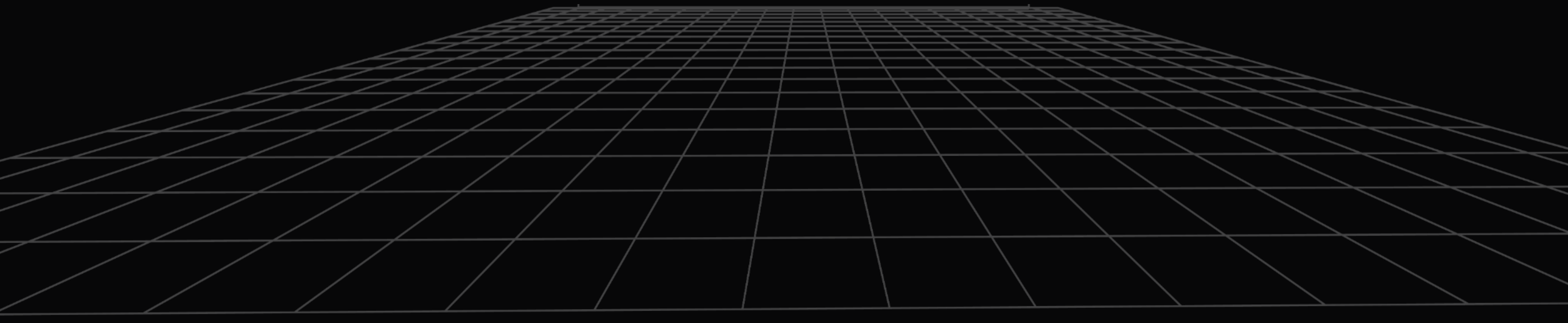
Продвинутый Go

ТИПЫ ДАННЫХ

Balun.Courses

ПРОВЕРЬ ЗАПИСЬ

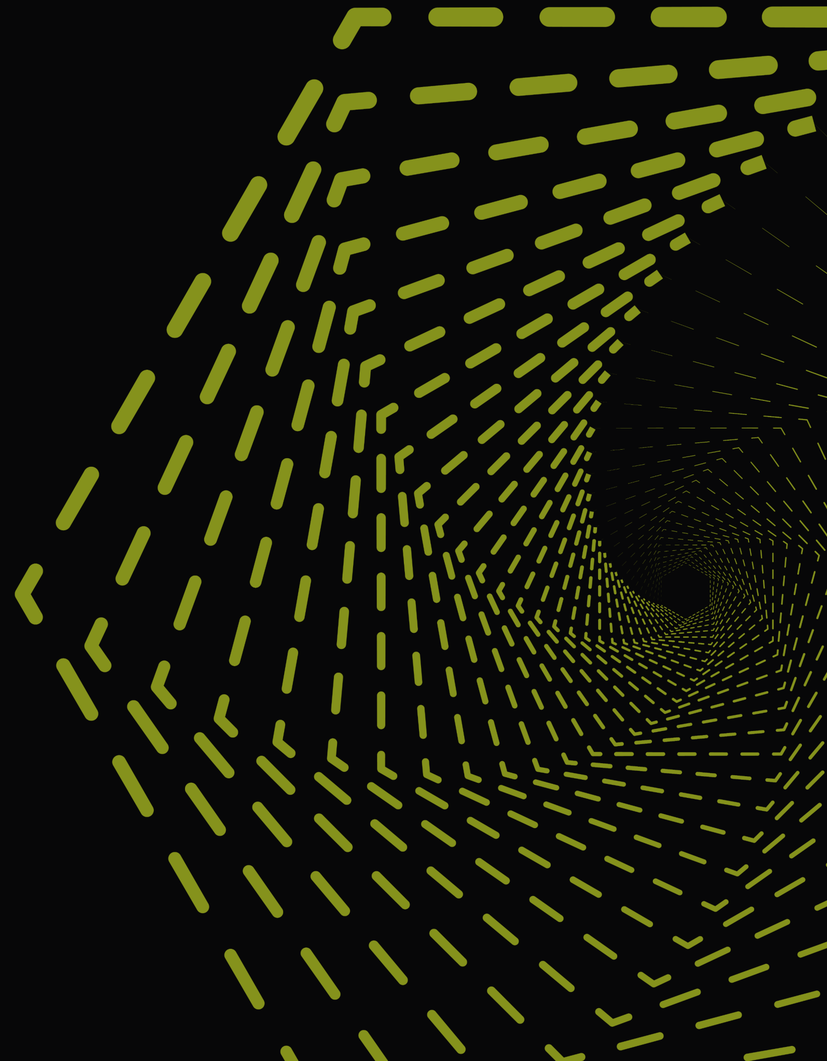


ПРАВИЛА ЗАНЯТИЯ

1. вопросы в чате можно задавать
   в любое время

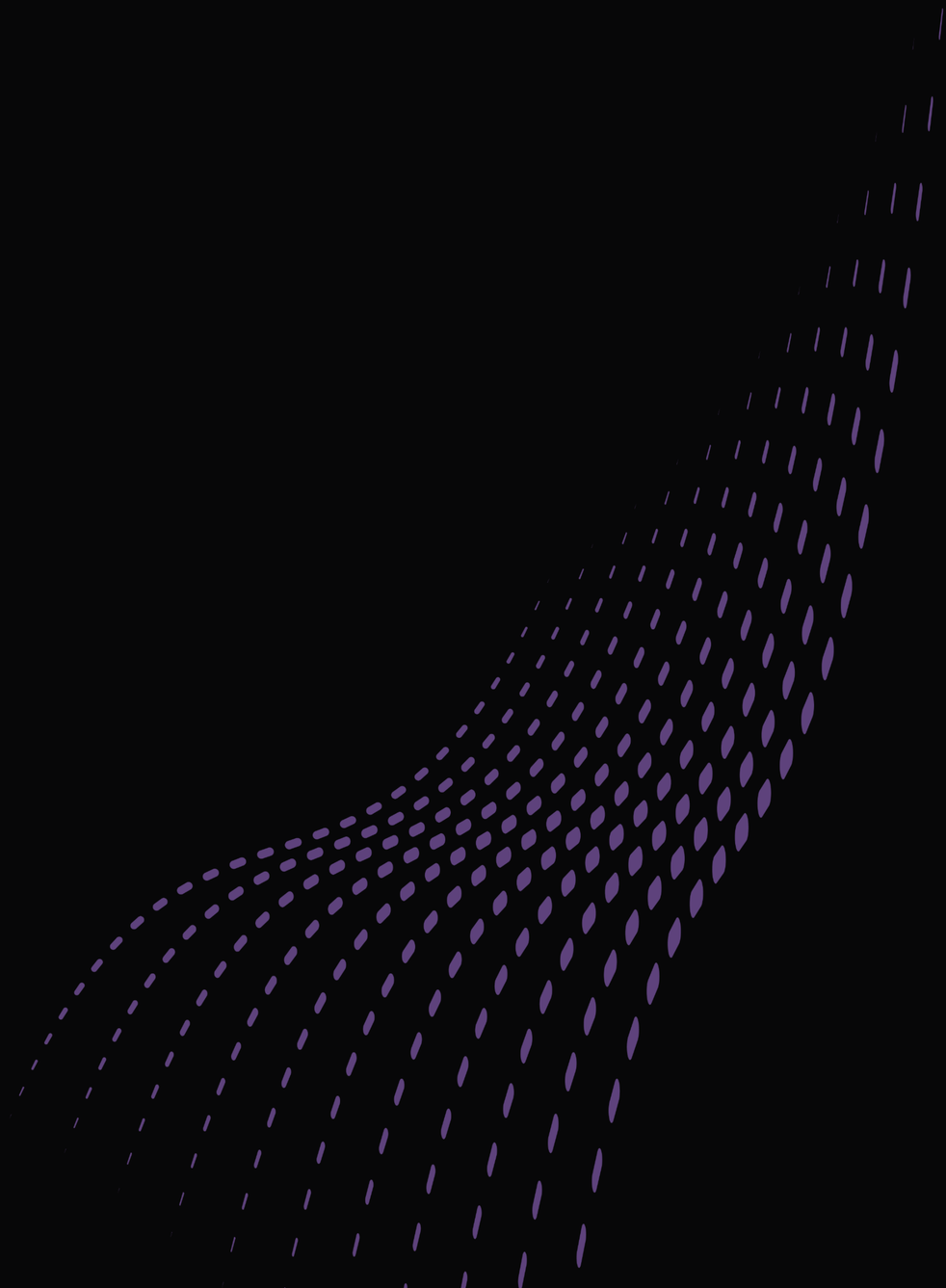
2. вопросы голосом задаем
   по поднятой руке в Zoom

3. ответы на вопросы будут
   в запланированных местах



ПЛАН ЛЕКЦИИ

1. Типы данных
2. Целочисленные типы данных
3. Указатели
4. Порядок следования байтов
5. Битовые операции
6. Битовые операции на практике



ТИПЫ ДАННЫХ

Terminal: question + ⌄ ⚙ –

ЧТО ТАКОЕ ТИП ДАННЫХ?

Terminal: deep_go

ТИП ДАННЫХ

Множество значений и операций, которые
могут быть применимы над этими значениями

ПРИМИТИВНЫЕ ТИПЫ ДАННЫХ

Логические

- `bool`

Строковые

- `string`

Числовые

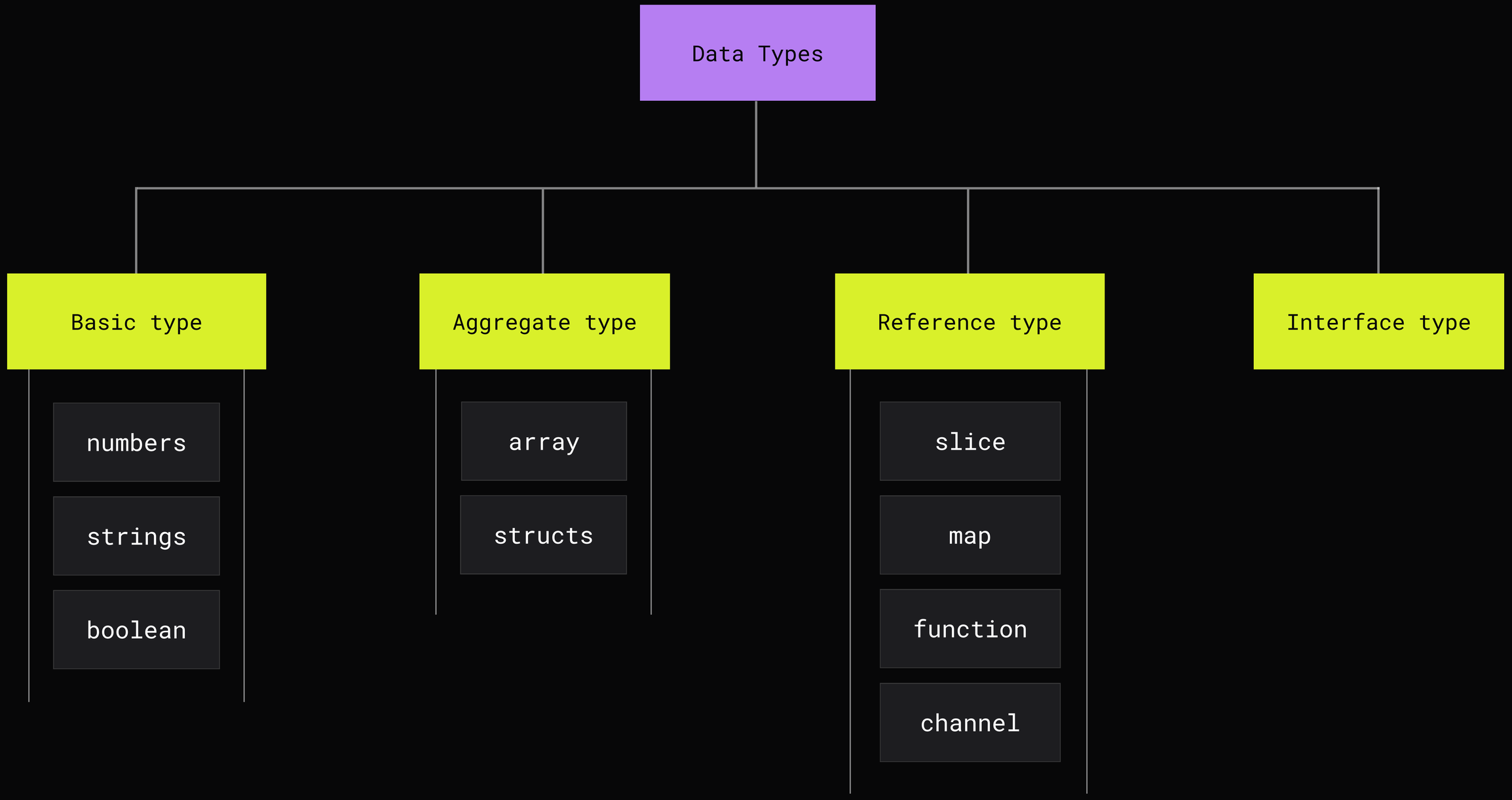
- `int8, uint8 (byte), int16, uint16, int32 (rune),uint32, int64, uint64, int, uint, uintptr`
- `float32, float64`
- `complex64, complex128`

СОСТАВНЫЕ ТИПЫ ДАННЫХ

1. **Указатели**
2. **Структуры**
3. **Функции**
4. **Контейнеры**
   - массивы
   - срезы
   - словари
5. **Каналы**
6. **Интерфейсы**

КАЖДЫЙ ТИП ИМЕЕТ НУЛЕВОЕ ЗНАЧЕНИЕ (ZERO VALUE),

которое можно рассматривать как значение типа по умолчанию


Data Types
Basic type
Aggregate type
Reference type
Interface type
numbers
strings
boolean
array
structs
slice
map
function
channel

Screenshot

Types whose values each is only hosted on one single memory block	Types whose values each may be hosted on multiple memory blocks
Solo Direct Value Part	Direct Part → Underlying Part
boolean types numeric types pointer types unsafe pointer types struct types array types	slice types map types channel types function types interface types string types

Terminal: question + ⚙ –

КАКОЙ РАЗМЕР У ОСНОВНЫХ ТИПОВ ДАННЫХ?

Бит – минимальная единица измерения информации,
которая принимает значение «ноль» или «один»

Байт (октет) – единица измерения информации,
которая состоит из восьми бит или двух тетрад

Машинное слово – основная единица информации,
которой оперирует процессор (размер машинного слова
равен архитектуре процессора, например 32 или 64
битная архитектура)

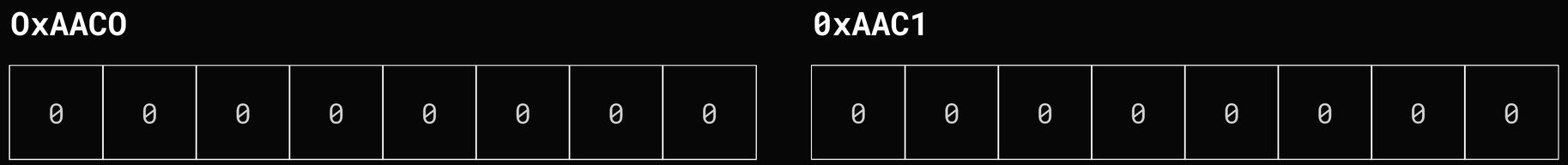
Screenshot

type	size in bytes
uint8, int8	1
uint16, int16	2
uint32, int32, float32	4
uint64, int64	8
float64, complex64	8
complex128	16
uint, int	implementation-specific, generally 4 on 32-bit architectures, and 8 on 64-bit architectures.
uintptr	implementation-specific, large enough to store the uninterpreted bits of a pointer value.

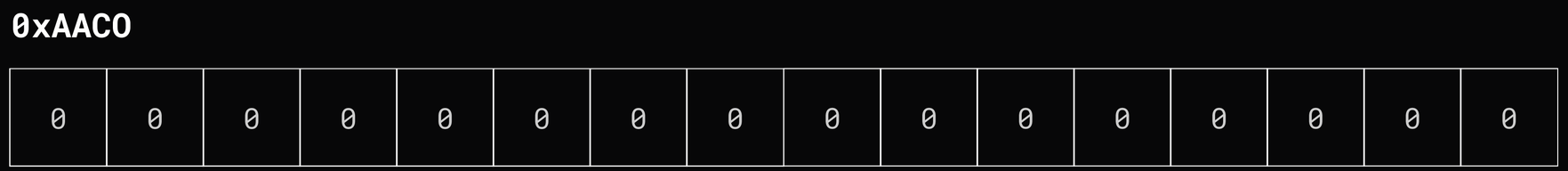
Terminal: question + ⌄ ⚙ –

ПОЧЕМУ РАЗМЕР ЛОГИЧЕСКОГО ТИПА ДАННЫХ (BOOL) 1 БАЙТ, А НЕ 1 БИТ?

Байтовая адресация

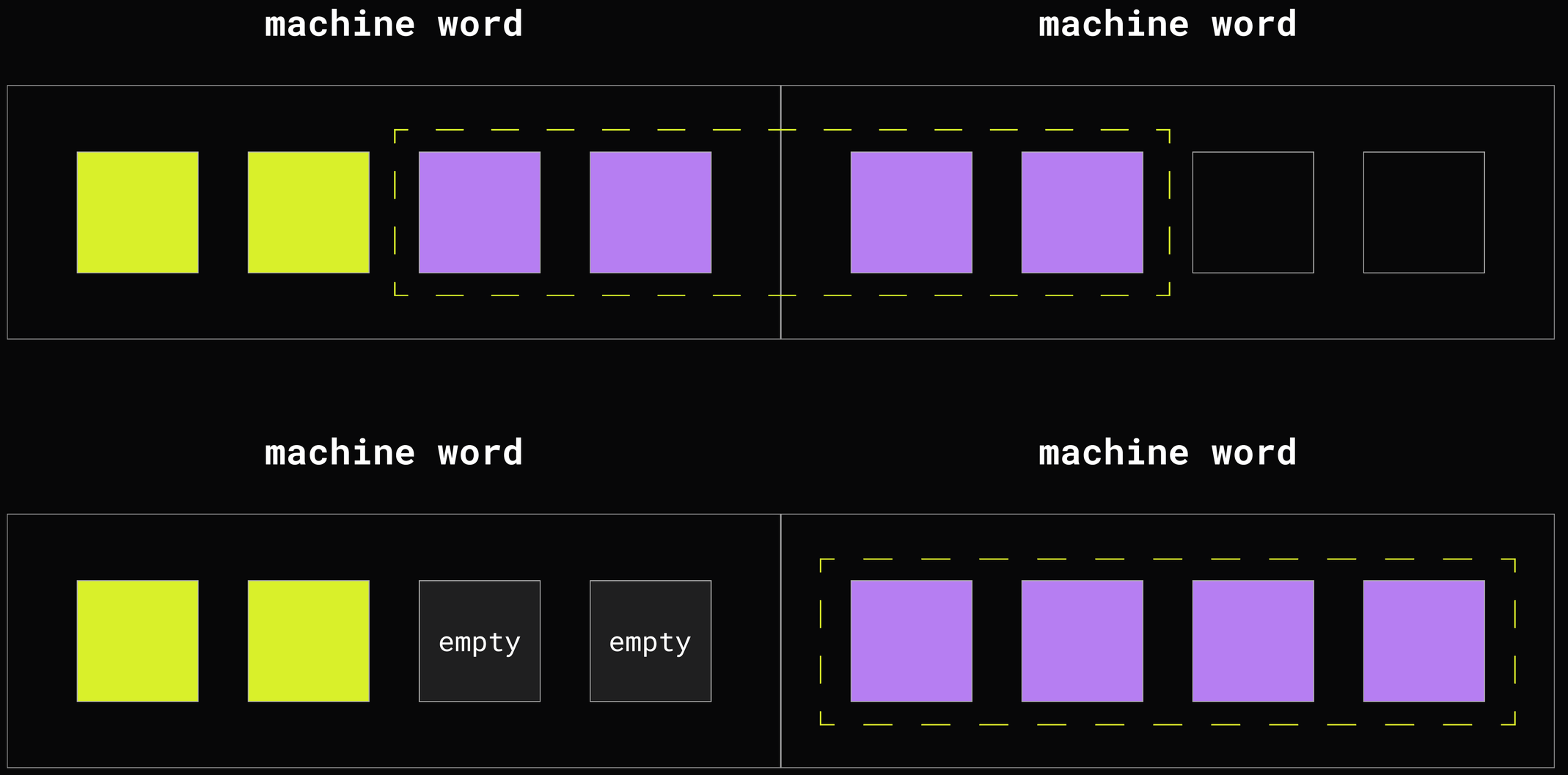


Словесная адресация



Terminal: question

НУЖНО ЛИ ВЫРАВНИВАТЬ ТИПЫ ДАННЫХ В ПАМЯТИ?

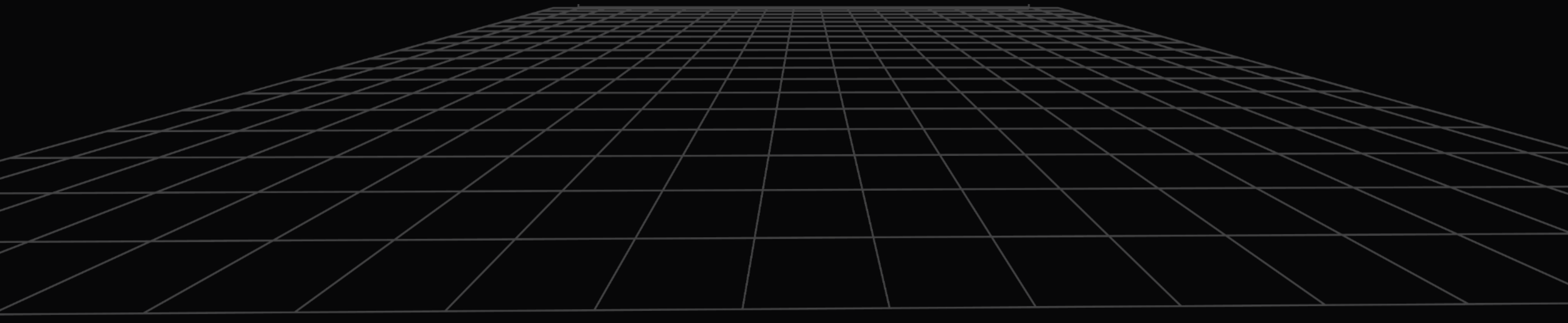

machine word
machine word
machine word
empty
empty
machine word

Screenshot

type	alignment guarantee
bool, uint8, int8	1
uint16, int16	2
uint32, int32	4
float32, complex64	4
arrays	depend on element types
structs	depend on field types
other types	size of a native word

FAQ

Типы данных



ЦЕЛОЧИСЛЕННЫЕ ТИПЫ ДАННЫХ

Terminal: deep_go

ПРЯМОЙ КОД ЧИСЛА

В десятичной системе счисления: **121**

В двоичной системе счисления: **0 1 1 1 1 0 0 1**

i 2^0 + 2^3 + 2^4 + 2^5 + 2^6 = 1 + 8 + 16 + 32 + 64 = 121

МОЖНО ЗАКОДИРОВАТЬ 2^N – 1 ЗНАЧЕНИЙ,

где N – количество бит

8 бит

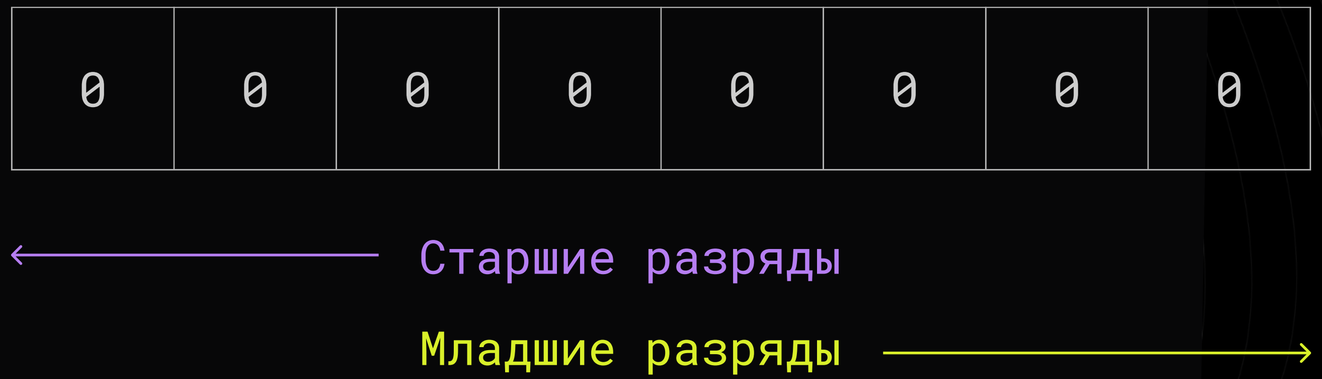
- 00000000 = 0
- 11111111 = 255

16 бит

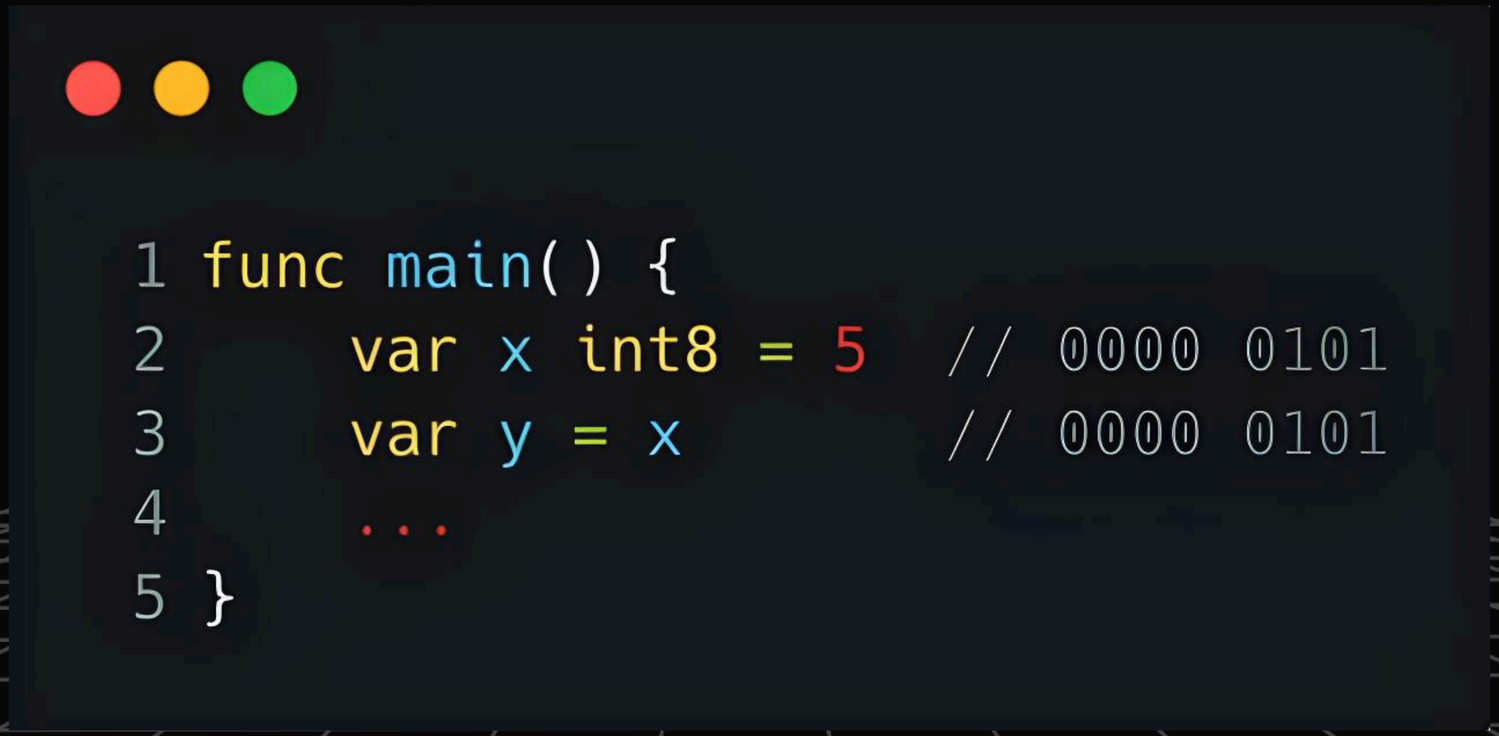
- 00000000 00000000 = 0
- 11111111 11111111 = 65535

РАЗРЯДЫ БИТОВ

Седьмой разряд *(крайний левый)* называется старшим, а нулевой разряд *(крайний правый)* – младшим. Наименьшему значению соответствует комбинация нулей, а наибольшему значению соответствует комбинация единиц



КОПИРОВАНИЕ ПЕРЕМЕННЫХ



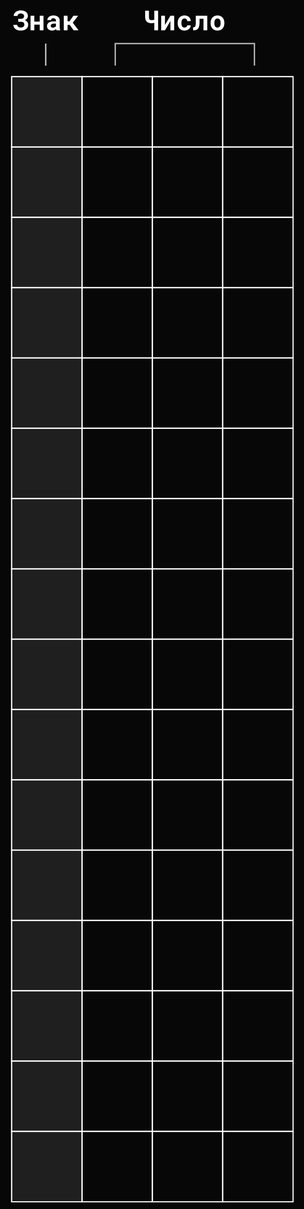
```
1 func main() {
2     var x int8 = 5   // 0000 0101
3     var y = x        // 0000 0101
4     ...
5 }
```

Terminal: question + ⚙ –

КАК БЫТЬ С ОТРИЦАТЕЛЬНЫМИ ЧИСЛАМИ?

0 – положительное

1 – отрицательное



Можно закодировать **2^N-1 – 1 (7)** положительных и **2^N-1 – 1 (7)** отрицательных чисел

Получилось два нуля –
положительный
и отрицательный, а также
-7 получился больше -1

Знак	Число			
0	1	1	1	7
0	1	1	0	6
0	1	0	1	5
0	1	0	0	4
0	0	1	1	3
0	0	1	0	2
0	0	0	1	1
0	0	0	0	0
1	0	0	0	0
1	0	0	1	-1
1	0	1	0	-2
1	0	1	1	-3
1	1	0	0	-4
1	1	0	1	-5
1	1	1	0	-6
1	1	1	1	-7

Не работает сложение

	0	0	1	0	2
+	1	0	1	0	-2
=	1	1	1	0	-6

Можно использовать **обратный код**, инвертируя все биты чисел

Знак	Число			
0	1	1	1	7
0	1	1	0	6
0	1	0	1	5
0	1	0	0	4
0	0	1	1	3
0	0	1	0	2
0	0	0	1	1
0	0	0	0	0
1	1	1	1	0
1	1	1	0	-1
1	1	0	1	-2
1	1	0	0	-3
1	0	1	1	-4
1	0	1	0	-5
1	0	0	1	-6
1	0	0	0	-7

Так работает

	0	0	1	0	2
+	1	1	0	1	-2
=	1	1	1	1	-0

Так не работает

	0	0	1	0	2
+	1	1	1	0	-1
=	0	0	0	0	0

Можно использовать **дополнительный код**, добавив 1 к обратному коду числа

Исчез **-0**, вместо него теперь можно добавить еще одно число

Можно закодировать **2^N-1 – 1** *(7)* положительных и **2^N-1***(8)* отрицательных чисел

Знак	Число			
0	1	1	1	7
0	1	1	0	6
0	1	0	1	5
0	1	0	0	4
0	0	1	1	3
0	0	1	0	2
0	0	0	1	1
0	0	0	0	0
1	1	1	1	-1
1	1	1	0	-2
1	1	0	1	-3
1	1	0	0	-4
1	0	1	1	-5
1	0	1	0	-6
1	0	0	1	-7
1	0	0	0	-8

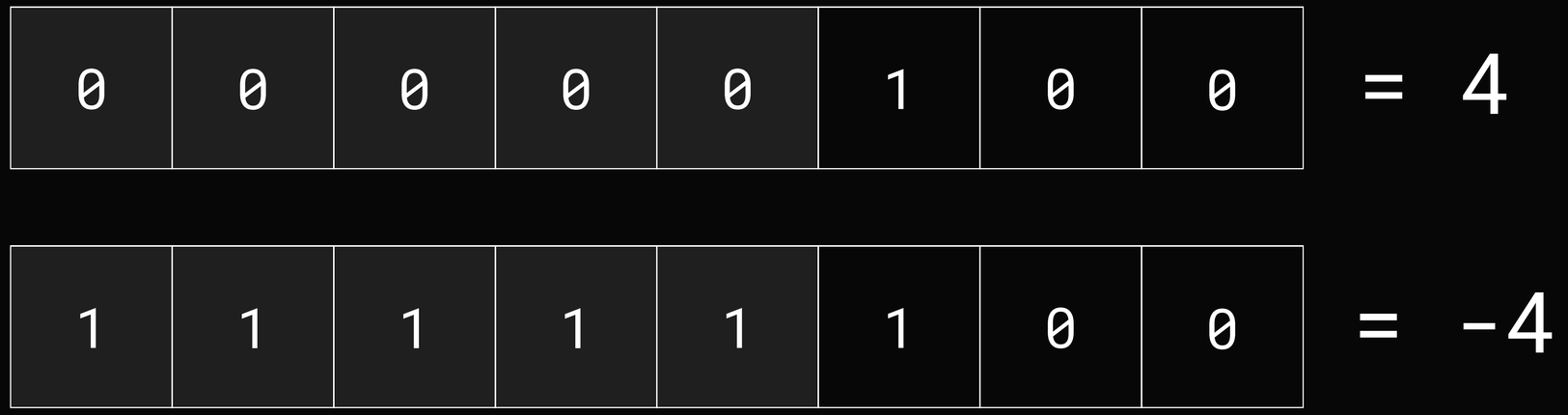
Так работает

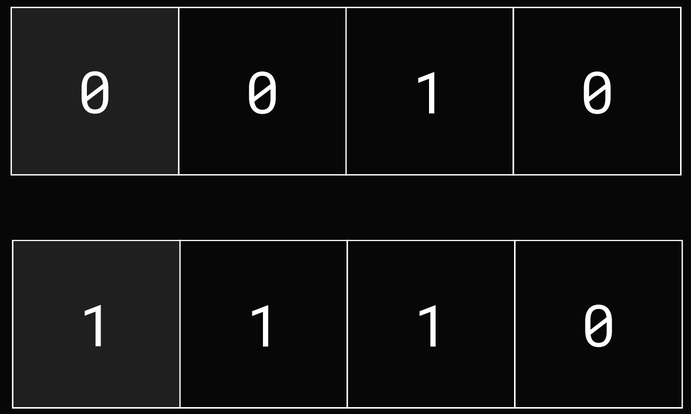
	0	0	1	0	2
+	1	1	1	0	-2
=	0	0	0	0	0

Так тоже работает

	0	0	1	0	2
+	1	1	1	1	-1
=	1	1	1	1	-1

Оставшиеся биты у положительных чисел заполняются нулями, а у отрицательных единицами



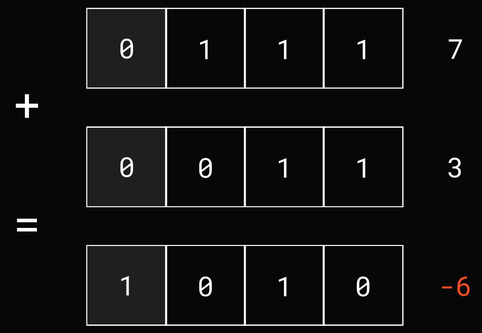


Знаковые числа компьютер может сравнивать лексикографически по первым битам числа

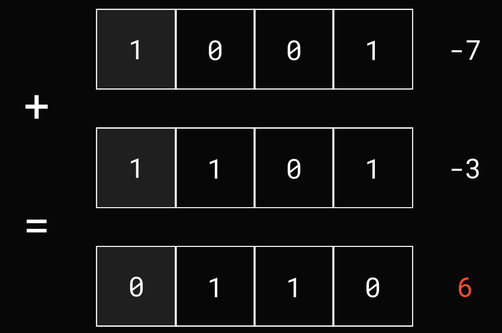
(получилась некая оптимизация)

ЧТО ПРОИЗОЙДЕТ, ЕСЛИ МЫ В ЧИСЛО ЗАПИШЕМ БОЛЬШЕ ИНФОРМАЦИИ,

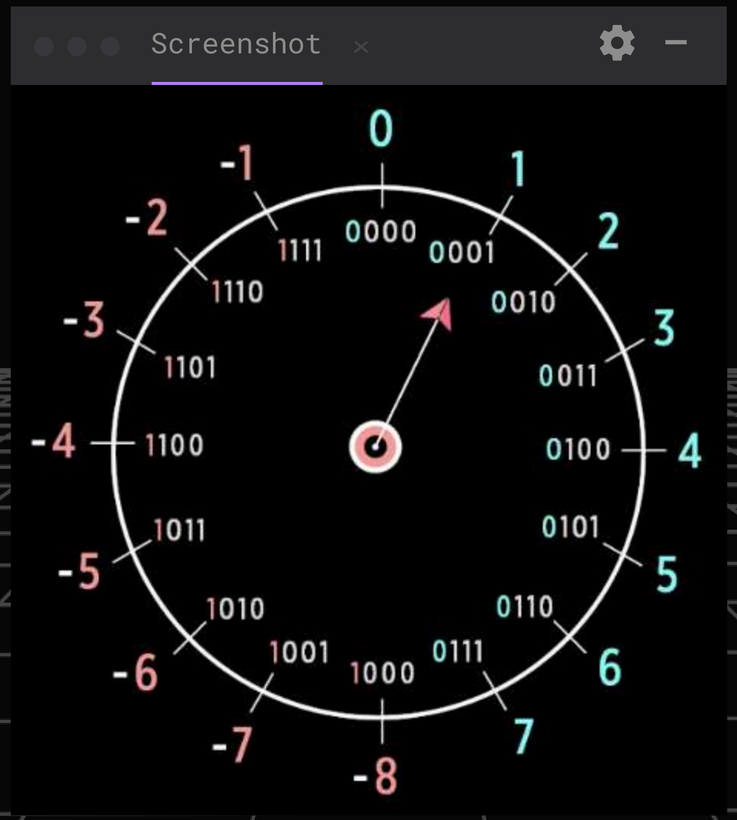
чем оно в себя может вместить?



Знак	Число			
0	1	1	1	7
0	1	1	0	6
0	1	0	1	5
0	1	0	0	4
0	0	1	1	3
0	0	1	0	2
0	0	0	1	1
0	0	0	0	0
1	1	1	1	-1
1	1	1	0	-2
1	1	0	1	-3
1	1	0	0	-4
1	0	1	1	-5
1	0	1	0	-6
1	0	0	1	-7
1	0	0	0	-8



НА КРУГЕ ВСЕ СТАНОВИТСЯ ПОНЯТНЕЕ



EXAMPLE

Int overflow

Неудачный запуск ракеты Ariane 5 в 1996 году произошел из-за переполнения, возникшего в результате преобразования 64-битного числа с плавающей точкой в 16-битное целое число со знаком

Terminal: question + ⚙ –

КАК ОБНАРУЖИТЬ ПЕРЕПОЛНЕНИЕ?

EXAMPLE

Overflow detection

ЦЕЛОЧИСЛЕННЫЕ ЛИТЕРАЛЫ

- **Двоичные** – используется префикс 0b или 0B *(например 0b100)*
- **Восьмеричные** – используется префикс 0 или 0o *(например 010)*
- **Шестнадцатеричные** – используется префикс 0x или 0X *(например 0xF)*
- **Мнимые** – используется суффикс i *(например 3i)*

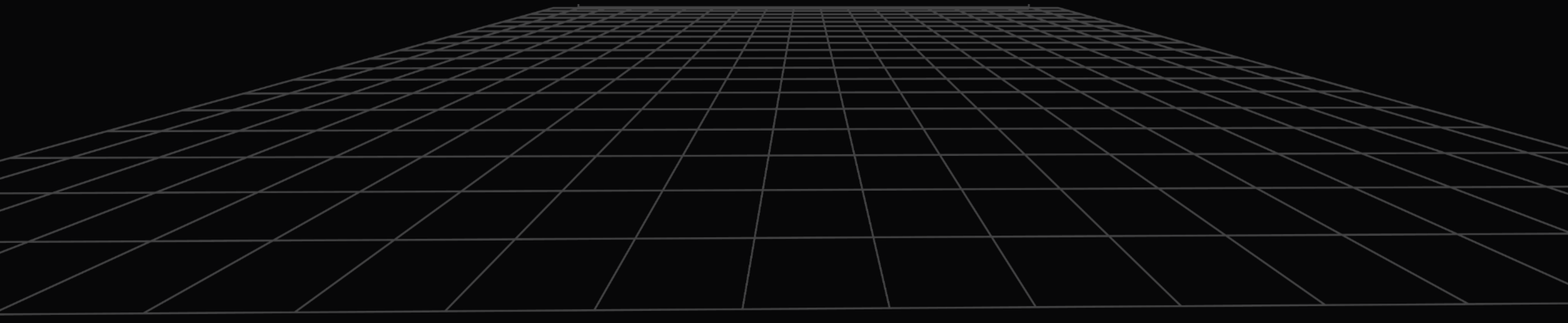
Еще можно использовать _ в качестве разделителя для удобства чтения, например **1_000_000**

EXAMPLE

Octal system

FAQ

Целочисленные типы данных



УКАЗАТЕЛИ

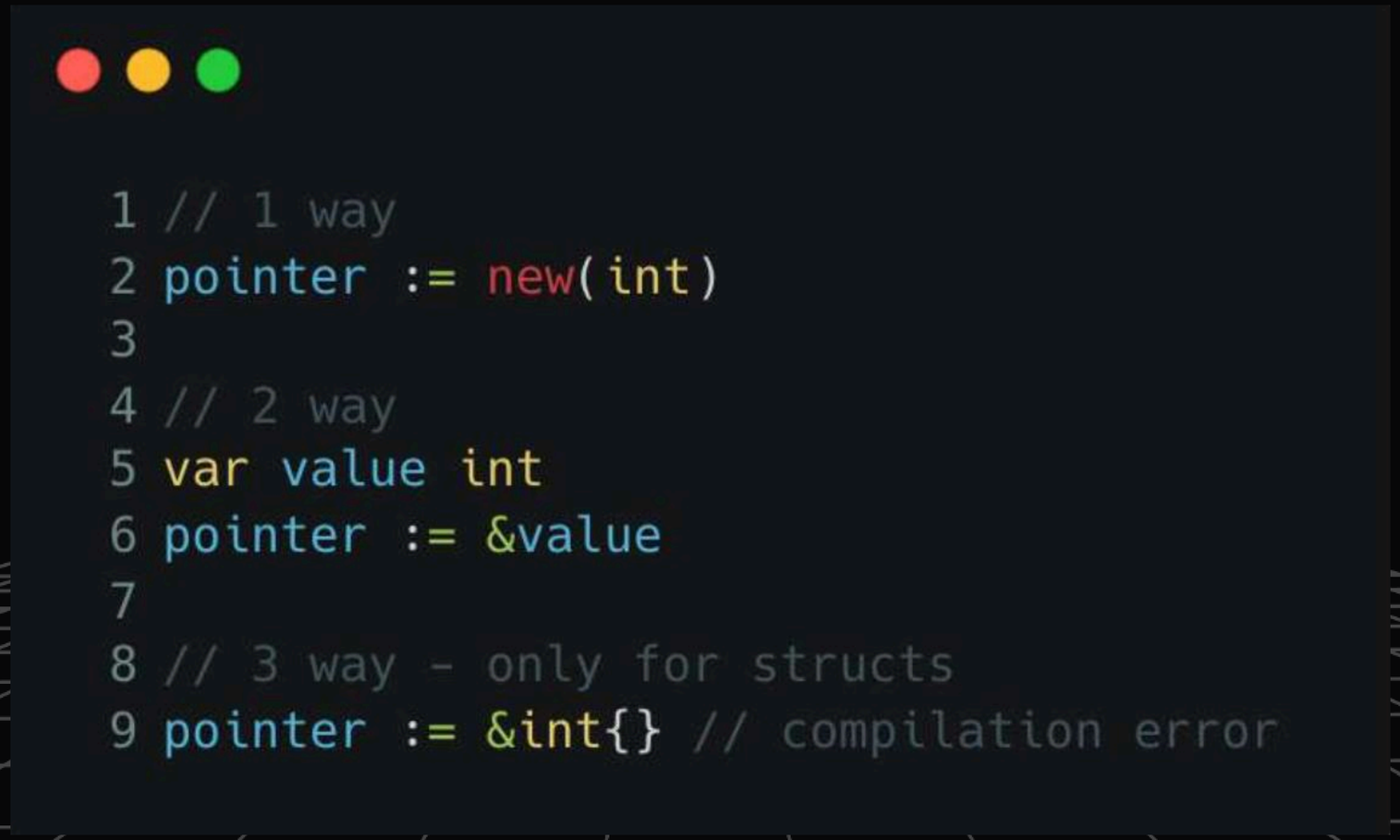
Terminal: deep_go

УКАЗАТЕЛЬ

Переменная, диапазон значений которой состоит
из адресов ячеек памяти или специального значения
– нулевого адреса (**nil**)

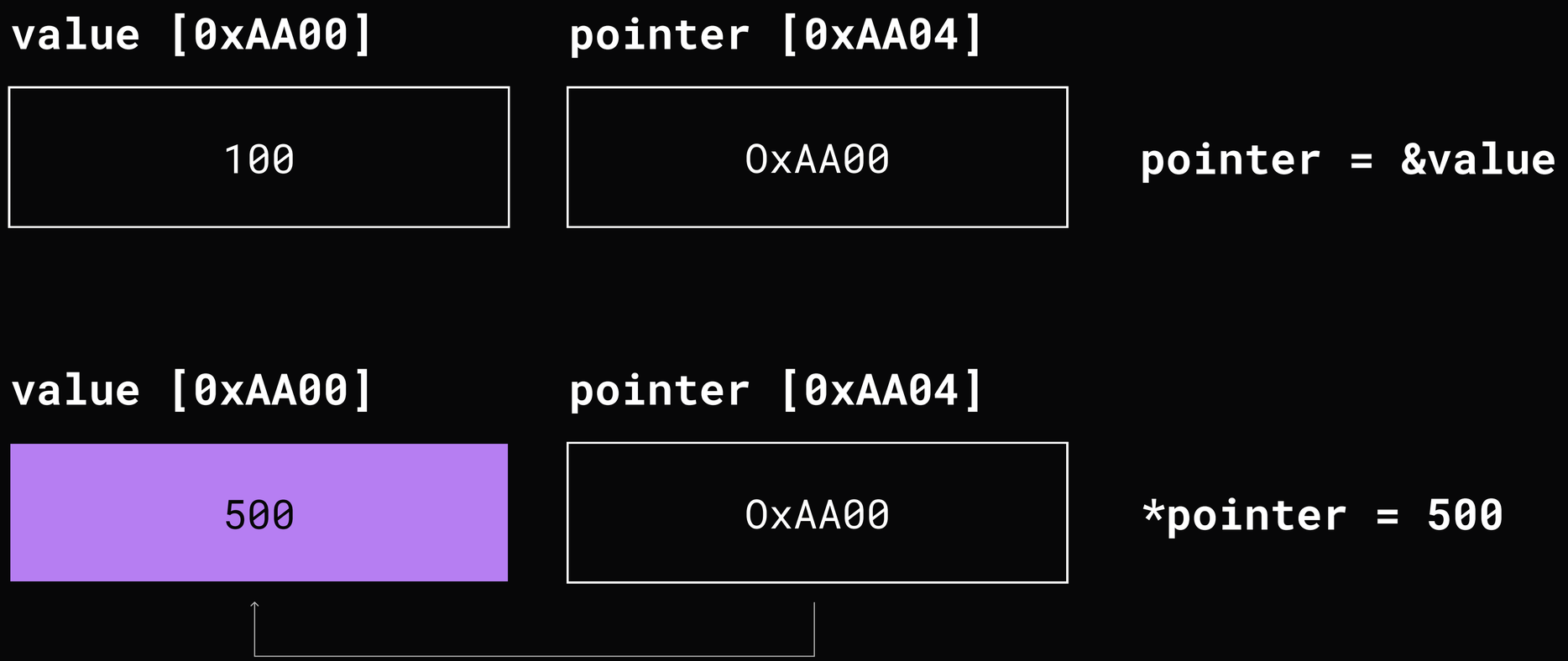
Terminal: question + ⚙ –

КАК ПОЛУЧИТЬ АДРЕС ПЕРЕМЕННОЙ?


// 1 way
pointer := new(int)
// 2 way
var value int
pointer := &value
// 3 way - only for structs
pointer := &int{} // compilation error

EXAMPLE

Pointer


value [0xAA00]
pointer [0xAA04]
100
0xAA00
pointer = &value
value [0xAA00]
pointer [0xAA04]
500
0xAA00
*pointer = 500
Разыменование

Terminal: question + ⚙ –

ЗАЧЕМ НУЖНЫ УКАЗАТЕЛИ?

EXAMPLE

Pointer with function

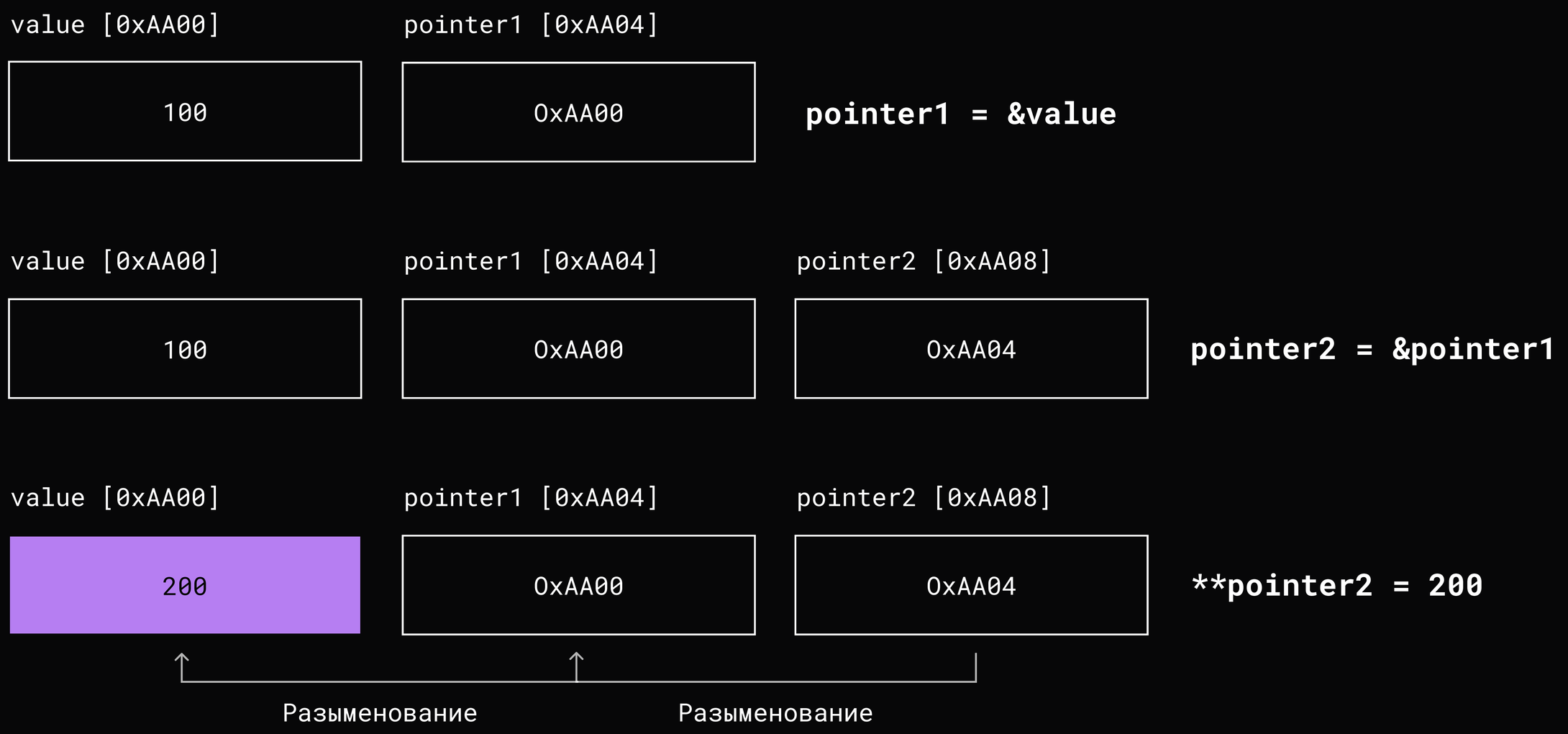
А также, чтобы вместо больших объектов передавать только адреса на них и чтобы строить структуры данных, основанные на указателях *(такие как связные списки, деревья и так далее)*

EXAMPLE

Pointer to pointer incorrect

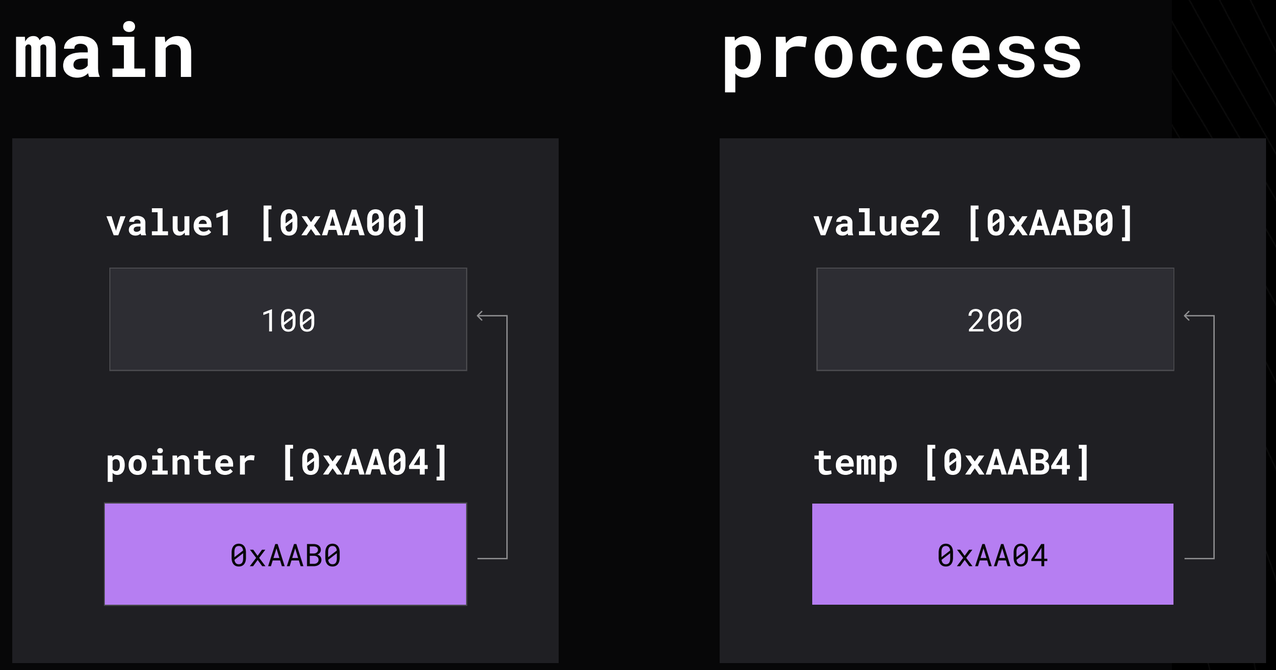

main
value1 [0xAA00]
100
pointer [0xAA04]
0xAA00
proccess
value2 [0xAAB0]
200
temp [0xAAB4]
0xAAB0

УКАЗАТЕЛИ НА УКАЗАТЕЛИ

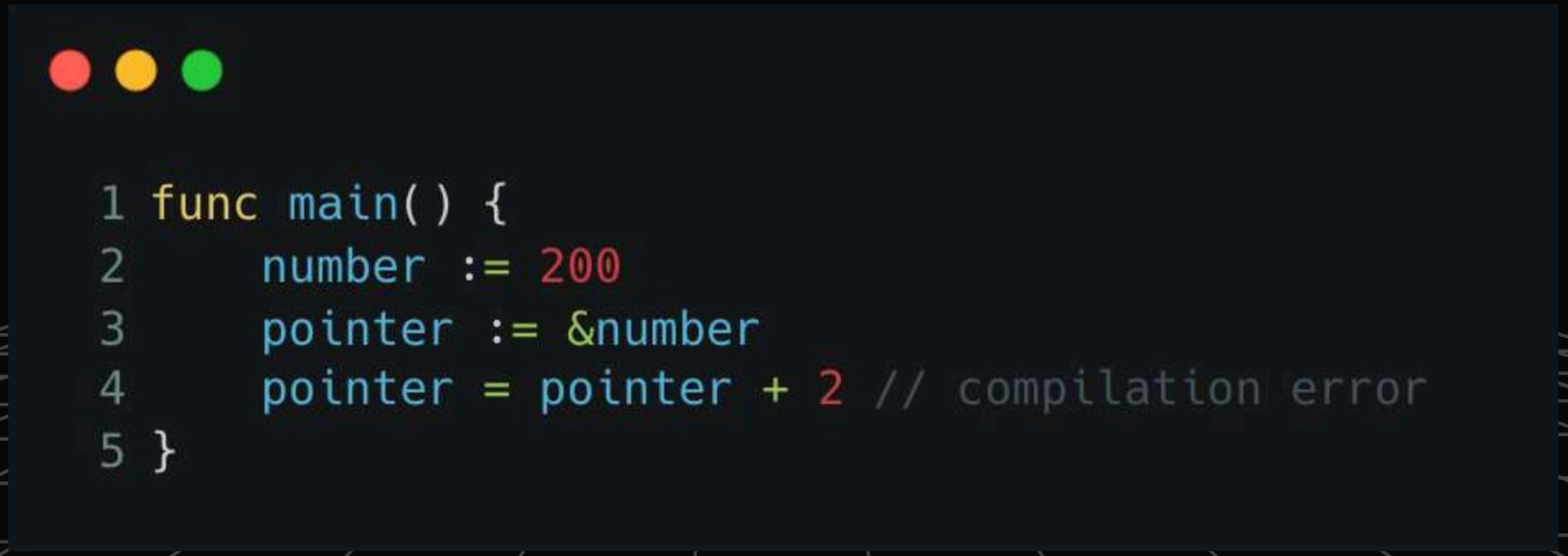


EXAMPLE

Pointer to pointer correct


main
value1 [0xAA00]
100
pointer [0xAA04]
0xAAB0
proccess
value2 [0xAAB0]
200
temp [0xAAB4]
0xAA04

В Go не поддерживается арифметика указателей,
но зато можно использовать пакет **unsafe**



```go
func main() {
    number := 200
    pointer := &number
    pointer = pointer + 2 // compilation error
}
```

POINTER

Pointer represents a pointer to an arbitrary type. There are four special operations available for type Pointer that are not available for other types:

- A pointer value of any type can be converted to a Pointer.
- A Pointer can be converted to a pointer value of any type.
- A uintptr can be converted to a Pointer.
- A Pointer can be converted to a uintptr.

Terminal: deep_go

UINTPTR

This is an unsigned integer type which is large enough to hold any pointer address. Therefore its size is platform dependent. It is just an integer representation of an address.

EXAMPLE

Unsafe pointer

value

EXAMPLE

Unsafe size of

We know that, for some cases, **the unsafe mechanism can help us write more efficient Go code. However, it is very easy to introduce some subtle bugs which have very low possibilities to produce when using the unsafe mechanism.** A program with these bugs may run well for a long time, but suddenly behave abnormally and even crash at a later time. Such bugs are very hard to detect and debug

Another big risk of using unsafe pointers comes from the fact that the unsafe mechanism is not protected by the Go compatibility guideline - **code depending on unsafe pointers works today could break since a later Go version**

Параметр компилятора динамического анализа **-gcflags=all=-d=checkptr** поддерживается начиная с **Go Toolchain 1.14**. При использовании этой опции некоторые (но не все) неправильные использования указателей будут обнаружены во время выполнения (как только такое неправильное использование будет обнаружено, возникнет паника)

EXAMPLE

Uintptr gc

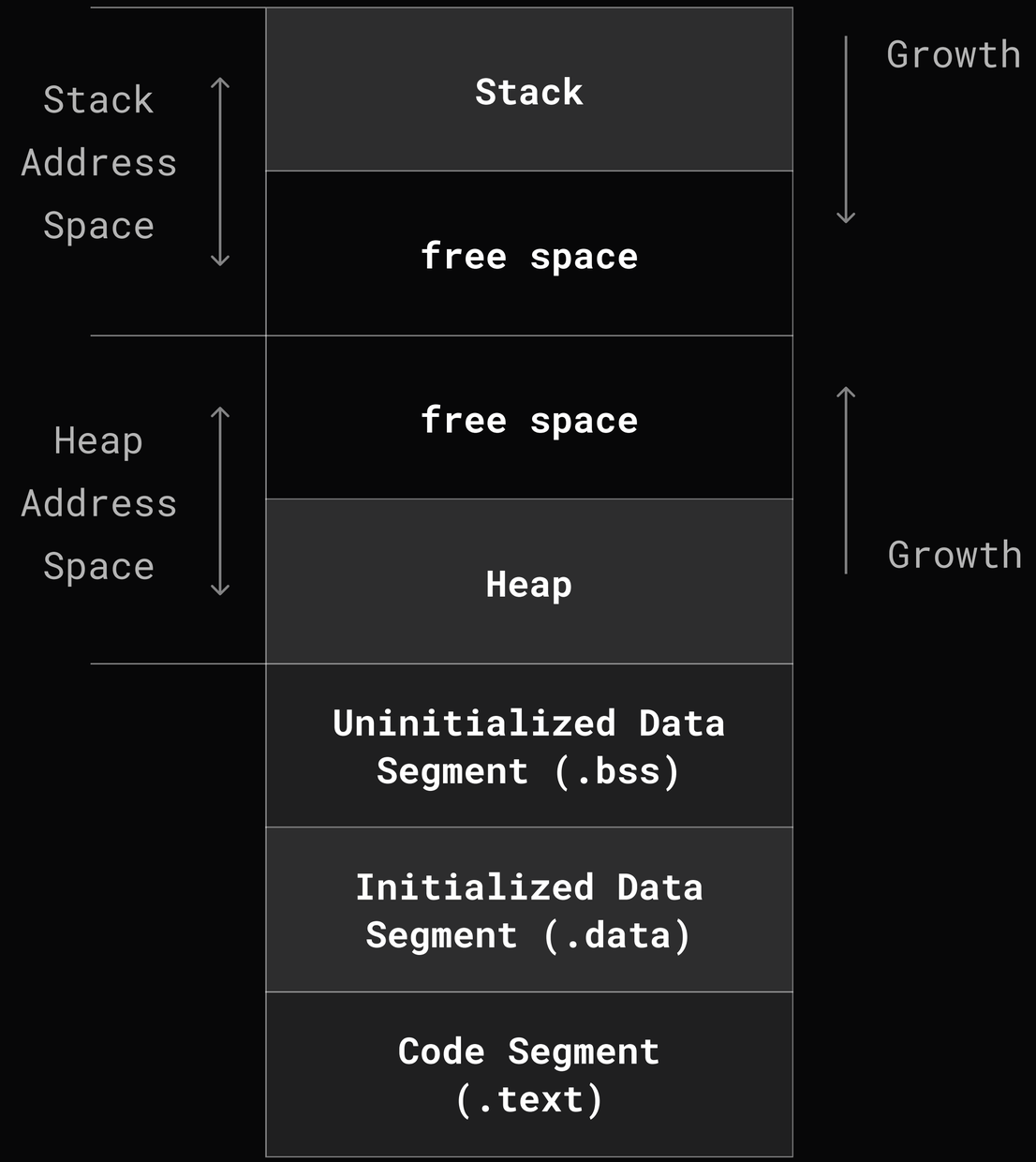
EXAMPLE

Uintptr stack grow

Each of non-nil safe and unsafe pointers references another value. However uintptr values don't reference any values, they are just plain integers, though often each of them stores an integer which can be used to represent a memory address.

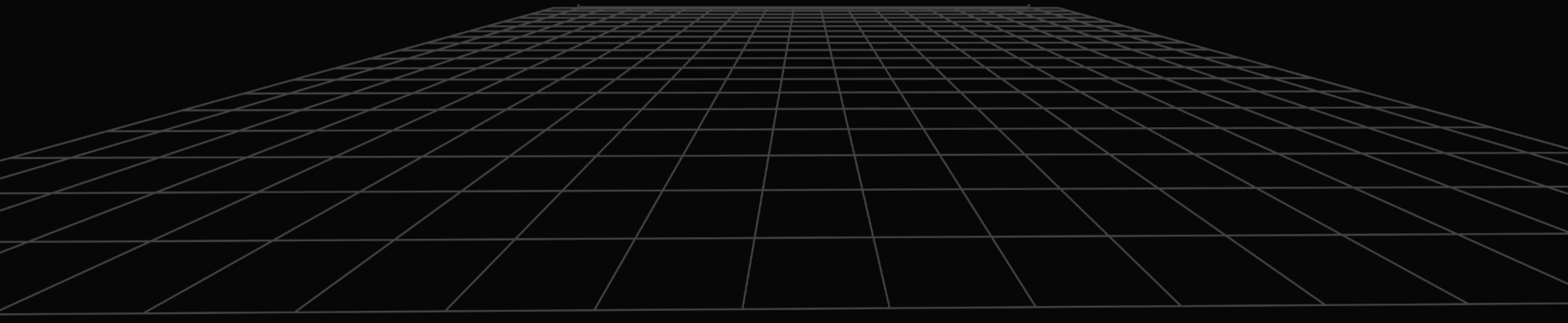
EXAMPLE

Unsafe analysis


Stack
Address
Space
Stack
free space
Growth
Heap
Address
Space
free space
Heap
Growth
Uninitialized Data
Segment (.bss)
Initialized Data
Segment (.data)
Code Segment
(.text)

FAQ

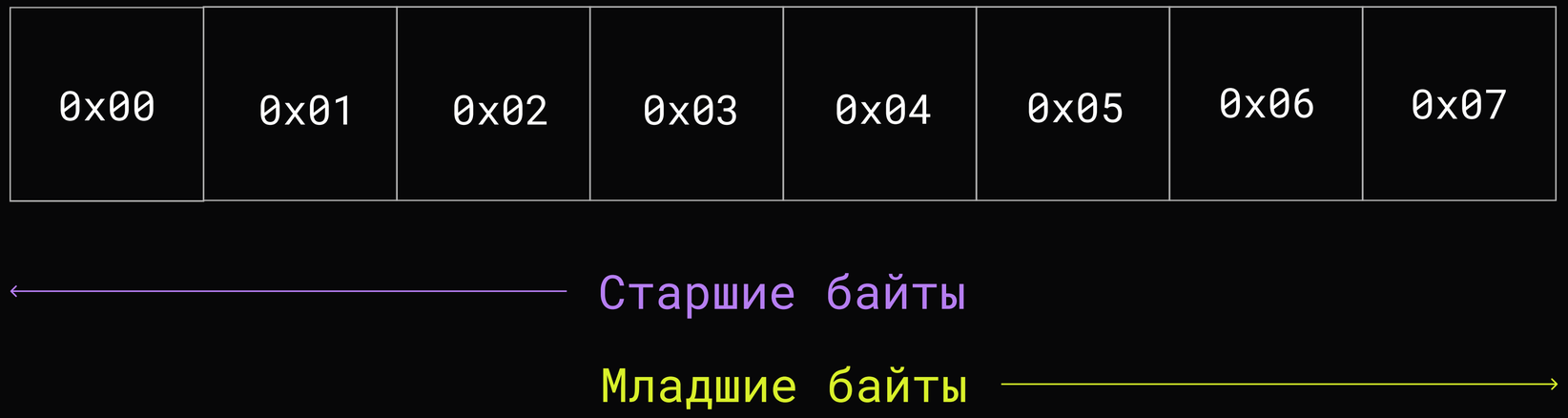
Указатели



ПЕРЕРЫВ 5 МИНУТ

ПОРЯДОК СЛЕДОВАНИЯ БАЙТОВ

Разряды байтов



EXAMPLE

Endianness

Terminal: deep_go ×

ENDIANNES

В том случае, если число не может быть
представлено одним байтом, имеет значение,
в каком порядке байты записываются в памяти
компьютера или передаются по линиям связи

Terminal: deep_go

BIG ENDIAN

Порядок от старшего к младшему - этот порядок подобен привычному порядку записи слева направо *(например, число сто двадцать три было бы записано при таком порядке как 123)*.

Этот порядок является стандартным для протоколов TCP/IP - он используется в заголовках пакетов данных и во многих протоколах более высокого уровня, поэтому такой порядок часто называют «сетевым порядком байтов»

Terminal: deep_go

LITTLE ENDIAN

Порядок от младшего к старшему - это обратный привычному порядку записи чисел *(например, число сто двадцать три было бы записано при таком порядке как 321)*. Этот порядок записи принят в памяти персональных компьютеров с процессорами архитектуры x86, в связи с чем иногда его называют «интеловским порядком байтов»

Big Endian

Little Endian

Порядок меняется только для байтов — биты
остаются в той же последовательности

Представьте, что вы отправляете письмо другу в другую страну, и в письме вы хотите сообщить ему о важной дате: 23 апреля 2023 года.

В России дату обычно записывают в формате день/месяц/год (23.04.2023), а в США — месяц/день/год (**04.23.2023**). Если ваш друг в США откроет письмо и увидит дату в вашем формате, он может неправильно понять, когда именно произойдет событие, особенно если день меньше **13**

Terminal: question + ⌄ ⚙ –

ЗАЧЕМ ПРИДУМАЛИ РАЗНЫЕ ПОРЯДКИ СЛЕДОВАНИЯ БАЙТ?

Современные процессоры x86 позволяют работать с одно, двух, четырёх и восьмибайтовыми операндами

В таком порядке следования байтов очень удобно то, что при увеличении размера (количества байтов) операнда, значение его первого байта неизменно: **3210 → 3210 0000**. При порядке от старшего к младшему значение изменилось бы, например: **0123 → 0000 0123**

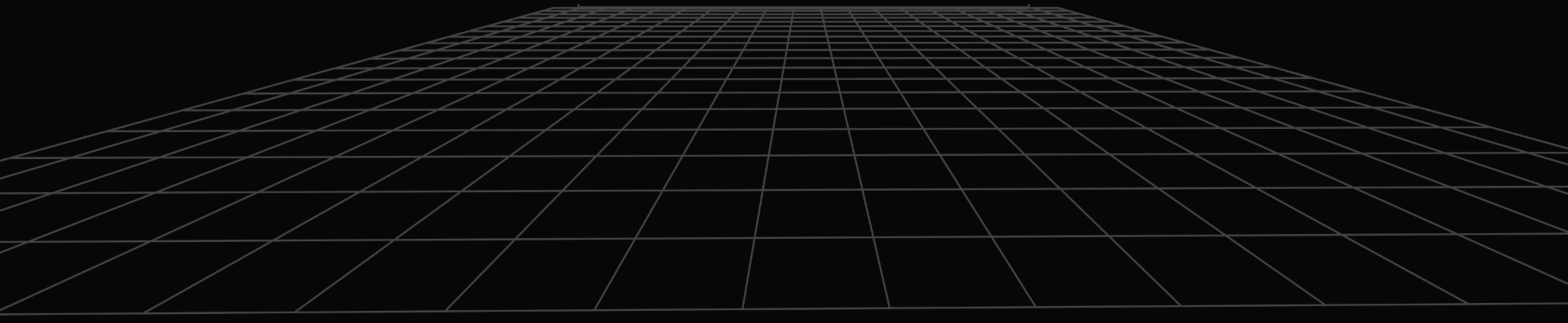
КАК ПРОВЕРИТЬ ПОРЯДОК СЛЕДОВАНИЯ БАЙТОВ?

EXAMPLE

Endianness check

FAQ

Порядок следования байтов



БИТОВЫЕ ОПЕРАЦИИ

Битовое И

```go
func main() {
    var x int8 = 3      // 0011
    var y int8 = 5      // 0101
    var result = x & y // 0001
}
```

Битовое ИЛИ

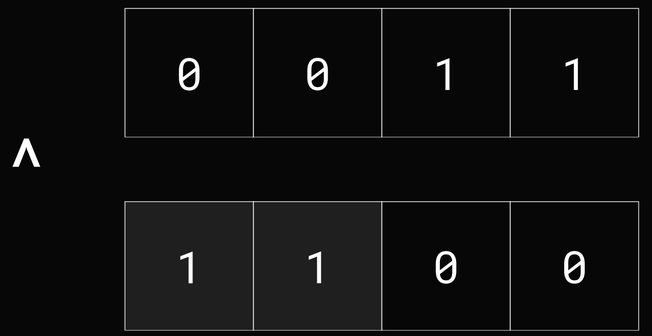
```go
func main() {
    var x int8 = 3      // 0011
    var y int8 = 5      // 0101
    var result = x | y // 0111
}
```

Исключающее ИЛИ (XOR)

```go
func main() {
    var x int8 = 3      // 0011
    var y int8 = 5      // 0101
    var result = x ^ y // 0110
}
```

Битовое НЕ

```go
func main() {
    var x uint8 = 3 // 0011
    var result = ^x // 1100
}
```

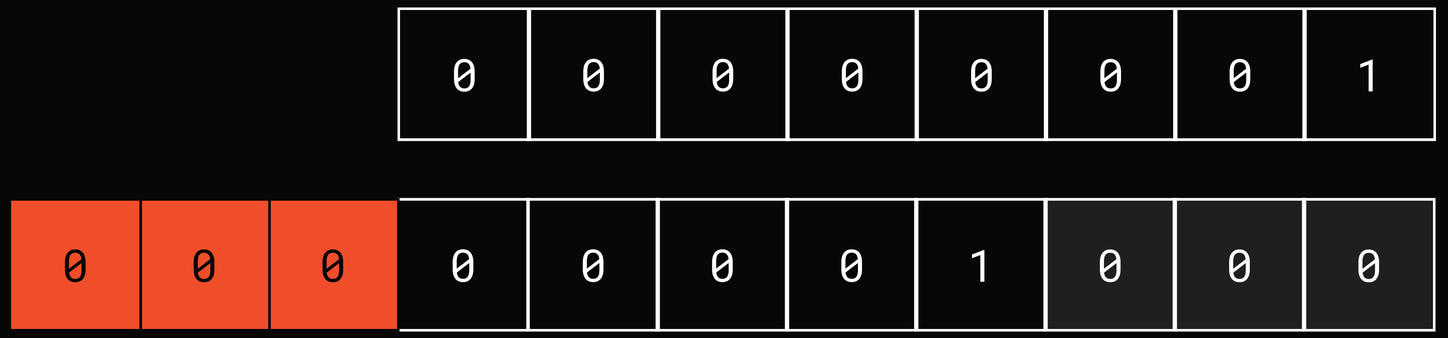


EXAMPLE

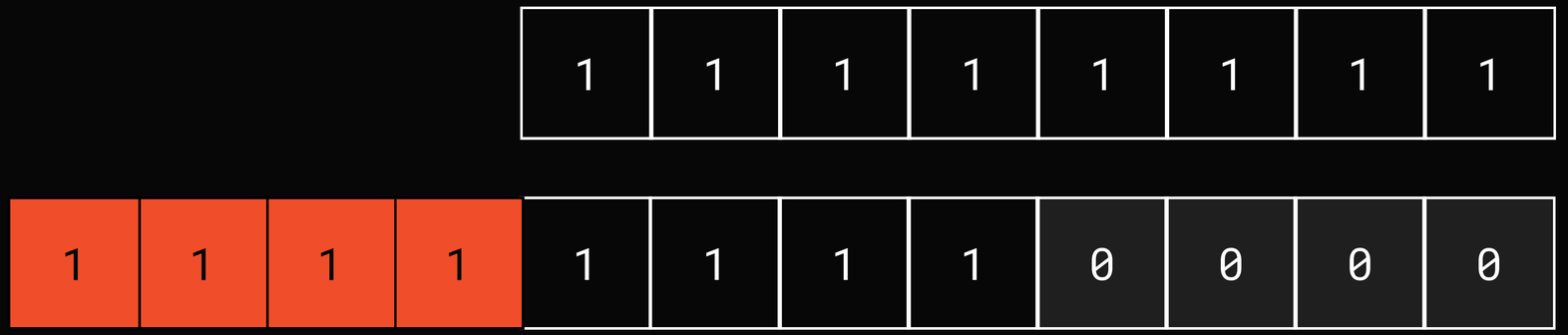
Inversion

Инверсия для отрицательных и положительных чисел работает по разному!

Битовые сдвиги



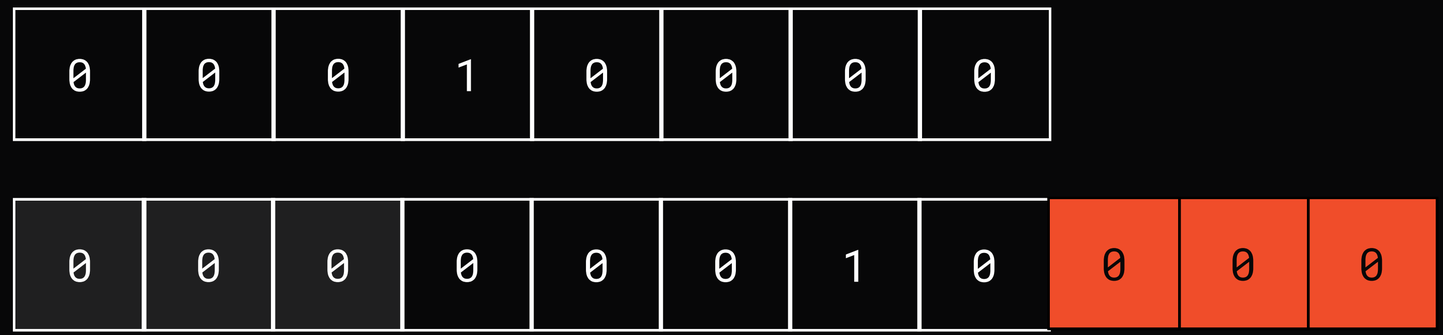
1 << 3



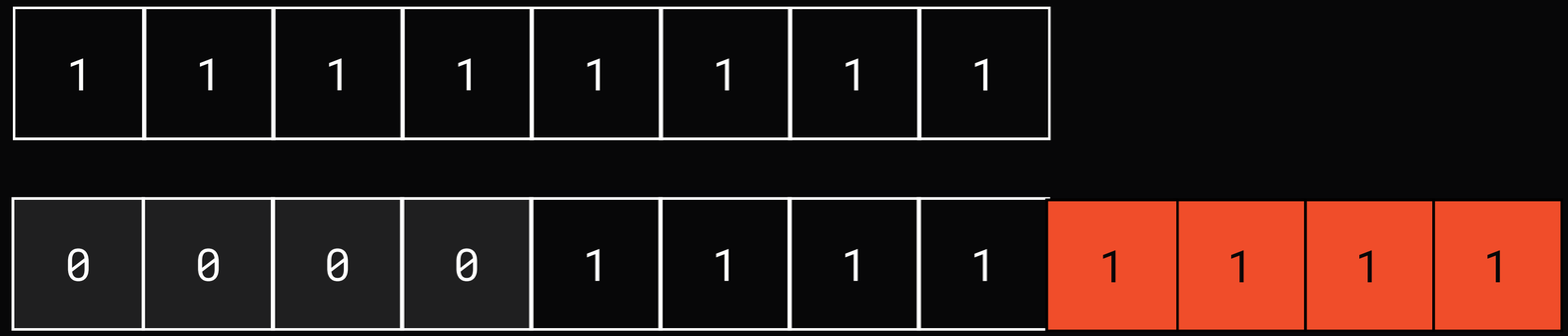
255 << 4

Битовые сдвиги

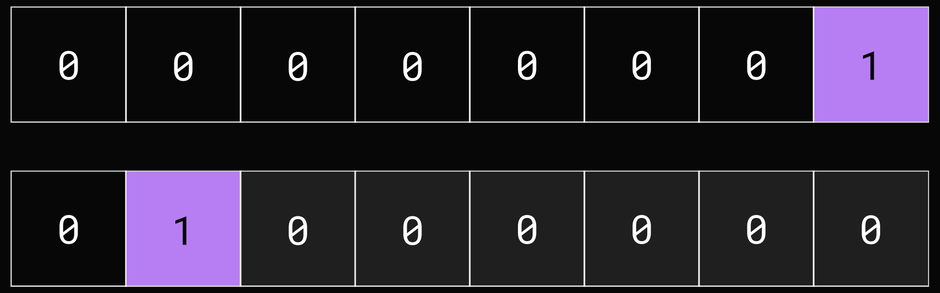
16 >> 3



255 >> 4

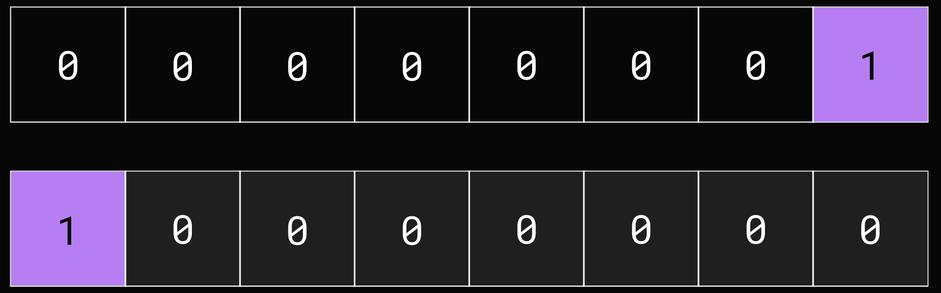


```go
func main() {
    var value int8 = 1
    value = value << 6
    fmt.Println(value) // 64
}
```



1<<6

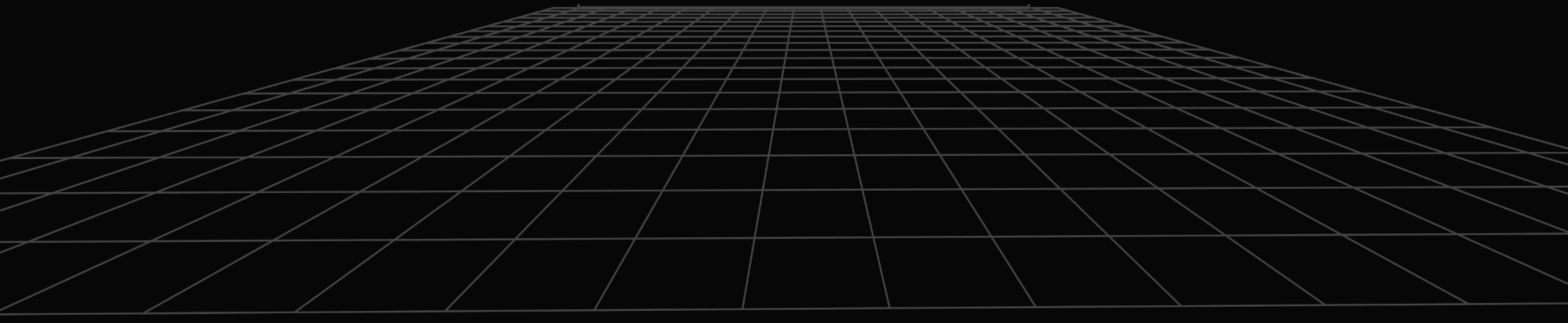
```go
func main() {
    var value int8 = 1
    value = value << 7
    fmt.Println(value) // -128
}
```



1<<7

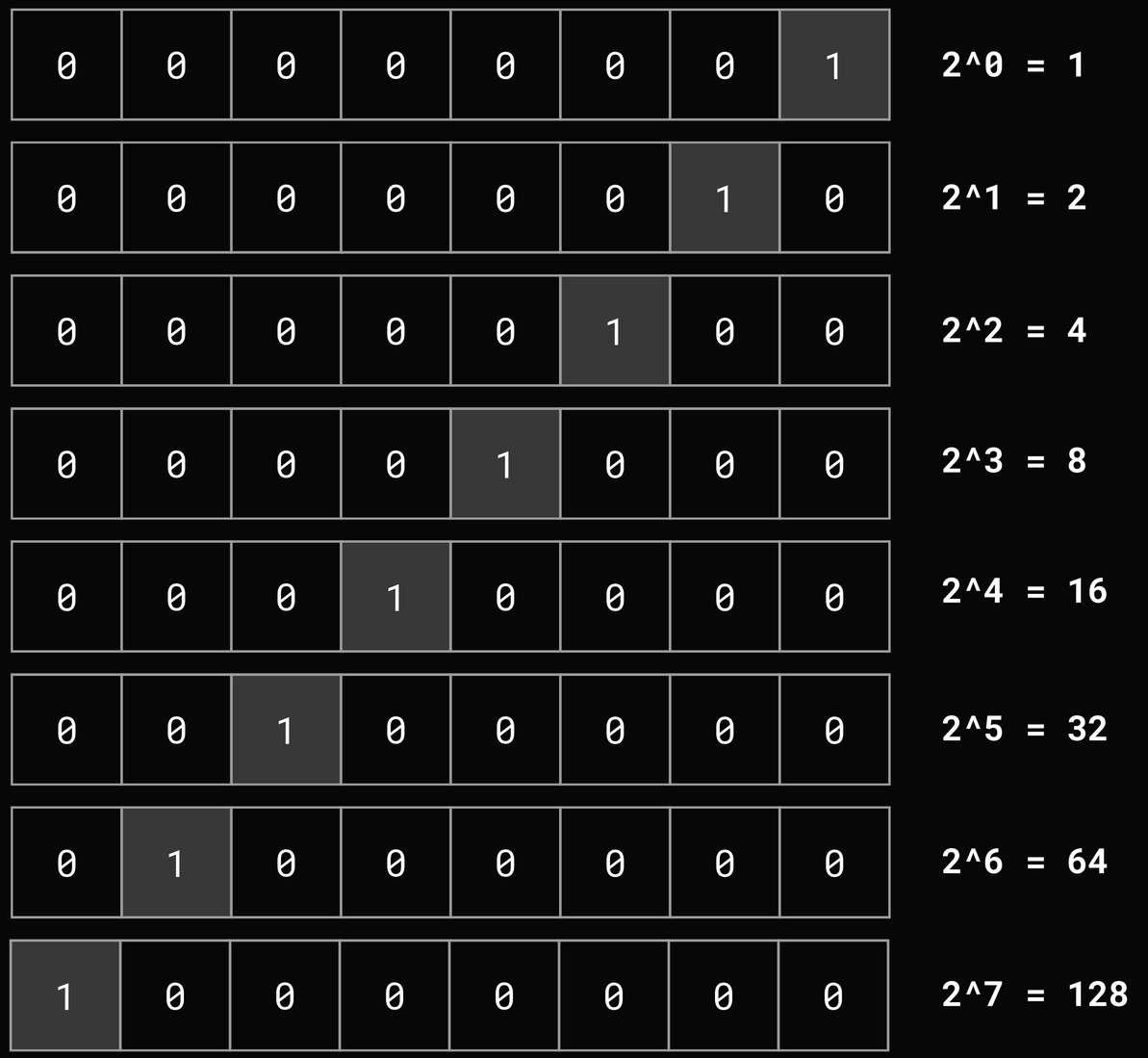
FAQ

Битовые операции



БИТОВЫЕ ОПЕРАЦИИ НА ПРАКТИКЕ

КАК УЗНАТЬ, ЯВЛЯЕТСЯ ЛИ ЧИСЛО СТЕПЕНЬЮ ДВОЙКИ?



Только одна единица в бинарном представлении числа

EXAMPLE

Power of two

Terminal: question + ⌄ ⚙ –

ЗАЧЕМ ЭТО НУЖНО ЗНАТЬ?

EXAMPLE

Bool mask

EXAMPLE

Bit mask

КАК НАЙТИ ЧИСЛО, КОТОРОЕ ВСТРЕЧАЕТСЯ ВО МНОЖЕСТВЕ ТОЛЬКО ОДИН РАЗ,

когда все остальные числа встречаются два раза?

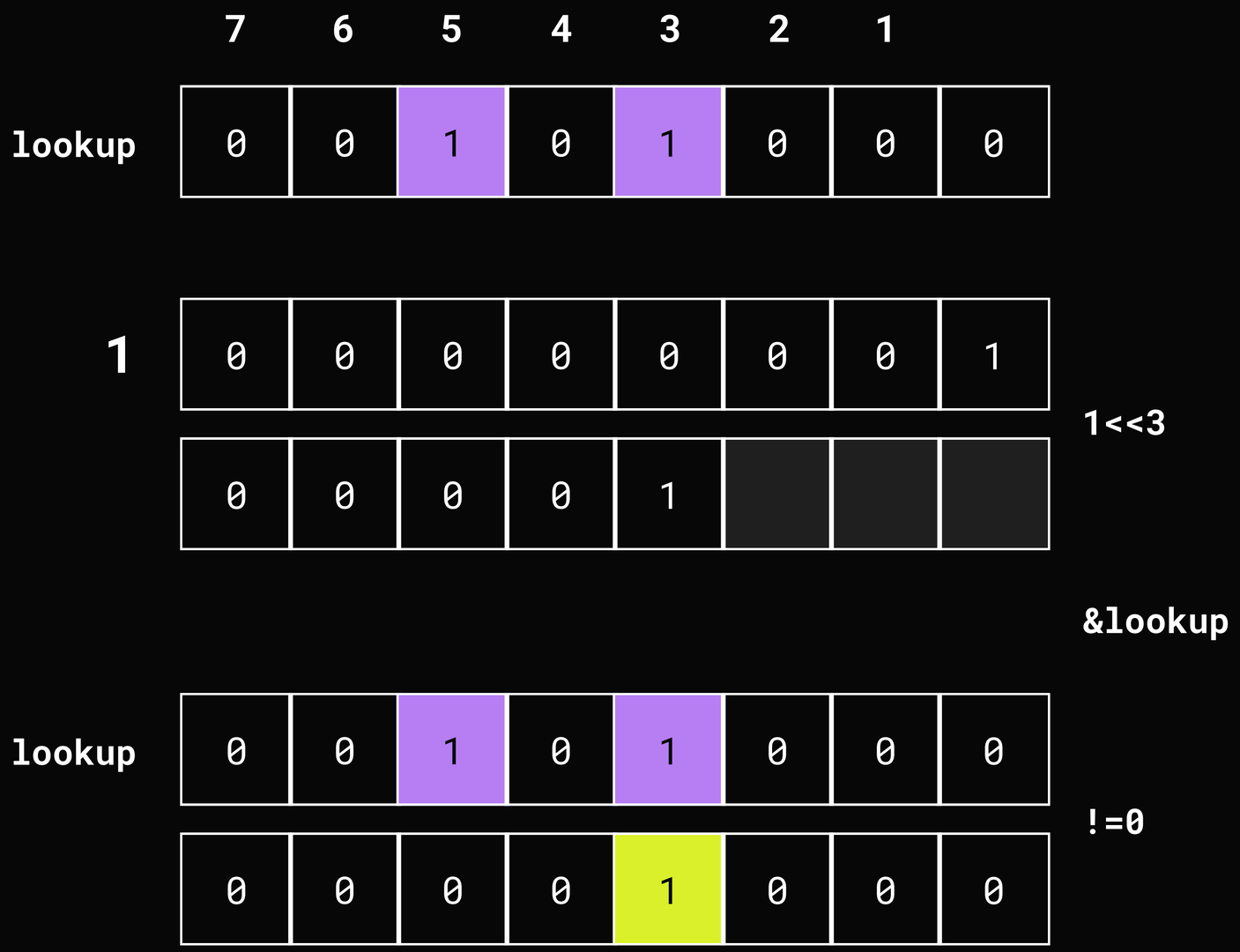
EXAMPLE

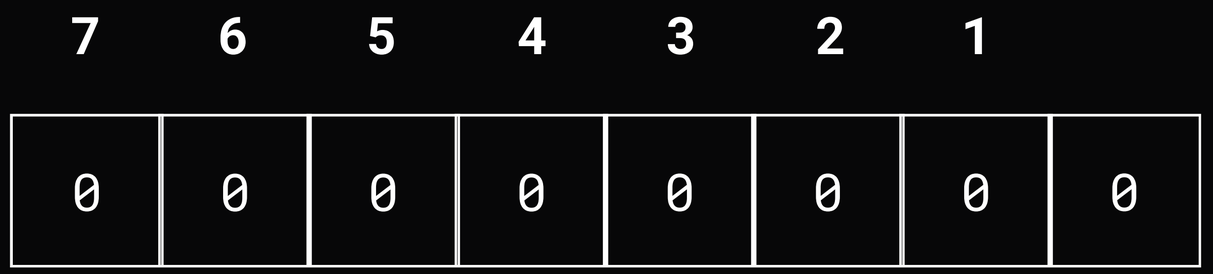
Find unique

КАК ПРОВЕРИТЬ, ЕСТЬ ЛИ ДУБЛИКАТЫ ВО МНОЖЕСТВЕ ЧИСЕЛ?

EXAMPLE

Duplicate check


7 6 5 4 3 2 1
lookup
0 0 1 0 1 0 0 0
1
0 0 0 0 0 0 0 1
1<<3
0 0 0 0 1
&lookup
lookup
0 0 1 0 1 0 0 0
!=0
0 0 0 0 1 0 0 0



Битовый карта *(bitset, bitmap, bit array)* — набор последовательно записанных двоичных разрядов, то есть последовательность битов

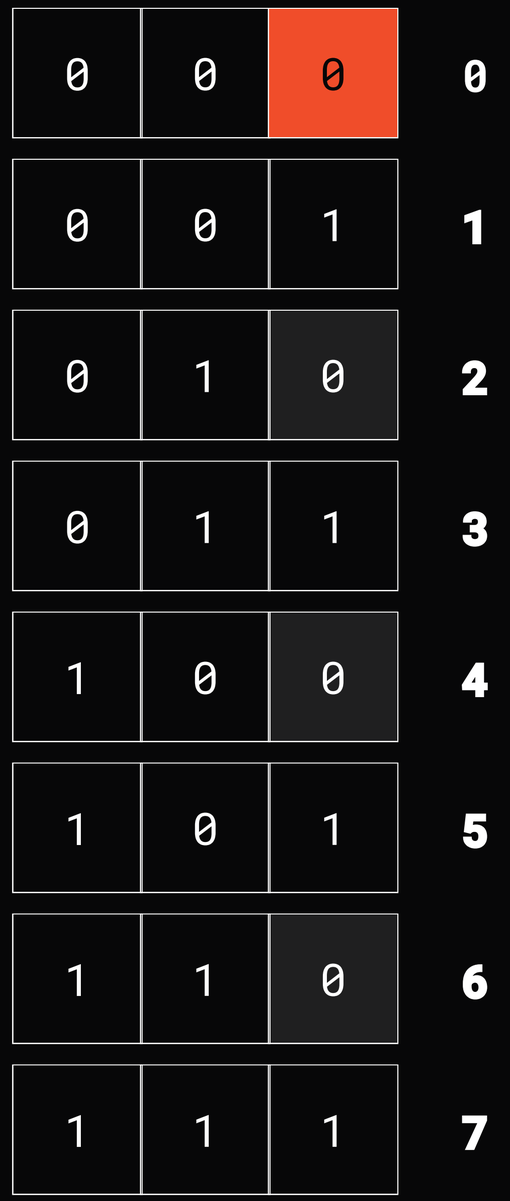
EXAMPLE

Bitmap index

КАК ПРОВЕРИТЬ ЯВЛЯЕТСЯ ЛИ ЧИСЛО ЧЕТНЫМ?

```go
func IsEven(number int) bool {
    return number & 1 == 0
}
```

У четных чисел последний бит равен нулю



EXAMPLE

Even test

Terminal: question + ⚙ –

КАК КОНВЕРТИРОВАТЬ БАЙТЫ, КИЛОБАЙТЫ, МЕГАБАЙТЫ?

EXAMPLE

Conversion test

n << m == n * 2^m

1КБ = 1024 Б	1 << 10 = 2^10	1024 Б
1МБ = 1024 КБ	1 << 20 = 2^20	1048576 Б
1ГБ = 1024 МБ	1 << 30 = 2^30	1073741824 Б
1ТБ = 1024 ГБ	1 << 40 = 2^40	1099511627776 Б
1ПБ = 1024 ТБ	1 << 50 = 2^50	1125899906842624 Б

Terminal: question

КАК ХРАНИТЬ И ПЕРЕДАВАТЬ IPV4 АДРЕС?

Terminal: deep_go

IPV4 (INTERNET PROTOCOL V4)

Адрес, записанный в 32-битном формате, имеет вид четырех 8-битных чисел *(минимум 0, максимум 255)*, которые разделены друг от друга точками *(пример - 172.16.255.2)*

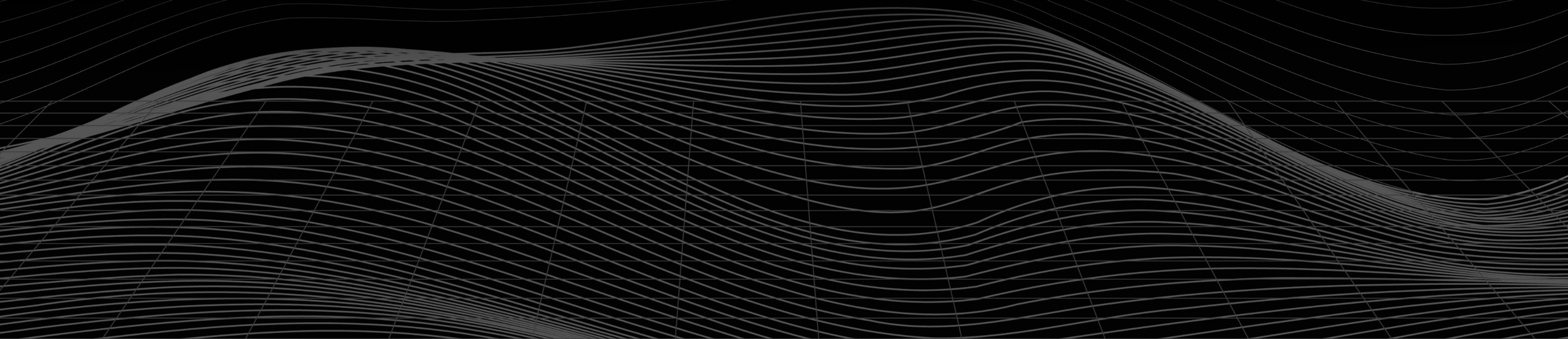
EXAMPLE

IPv4 address

Если вы используете битовые операции
– **скрывайте их за понятными абстракциями!**

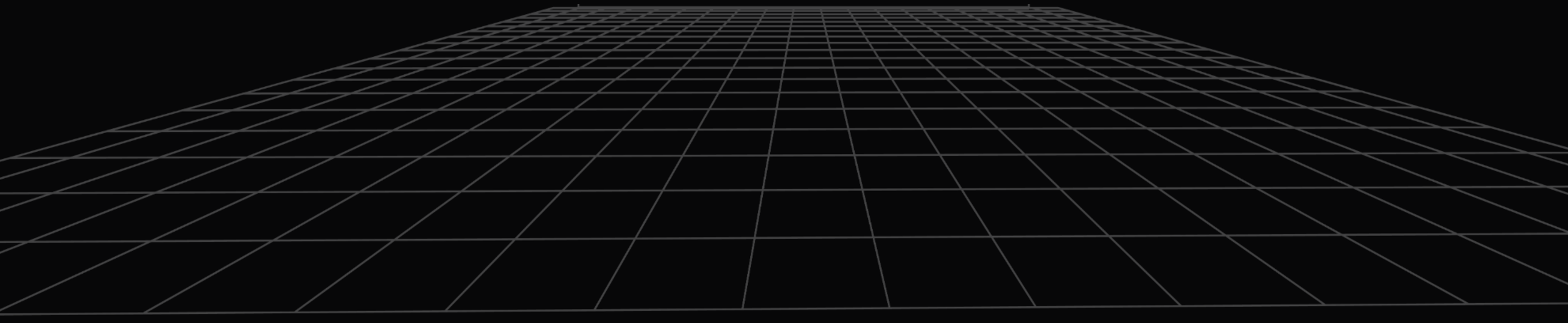
EXAMPLE

Bit wrappers



FAQ

Битовые операции на практике



ПОЖАЛУЙСТА, ЗАПОЛНИ ОПРОС О ЗАНЯТИИ

Ссылка в чате и в группе участников