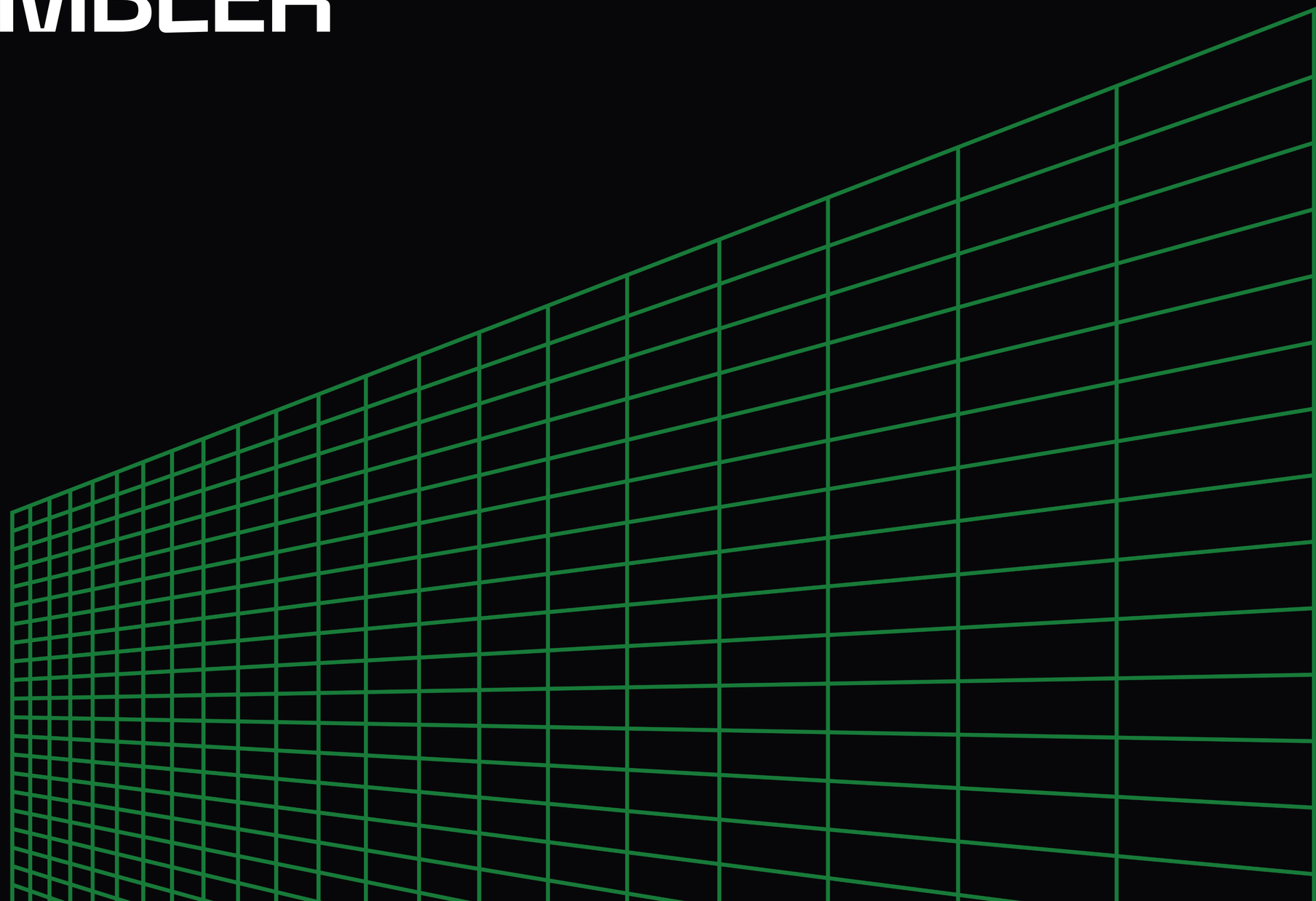
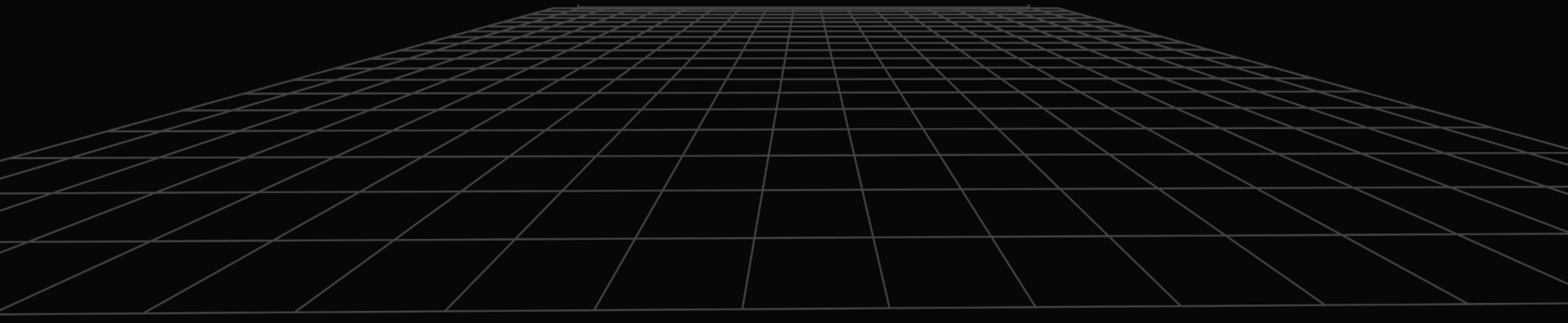
GO ASSEMBLER



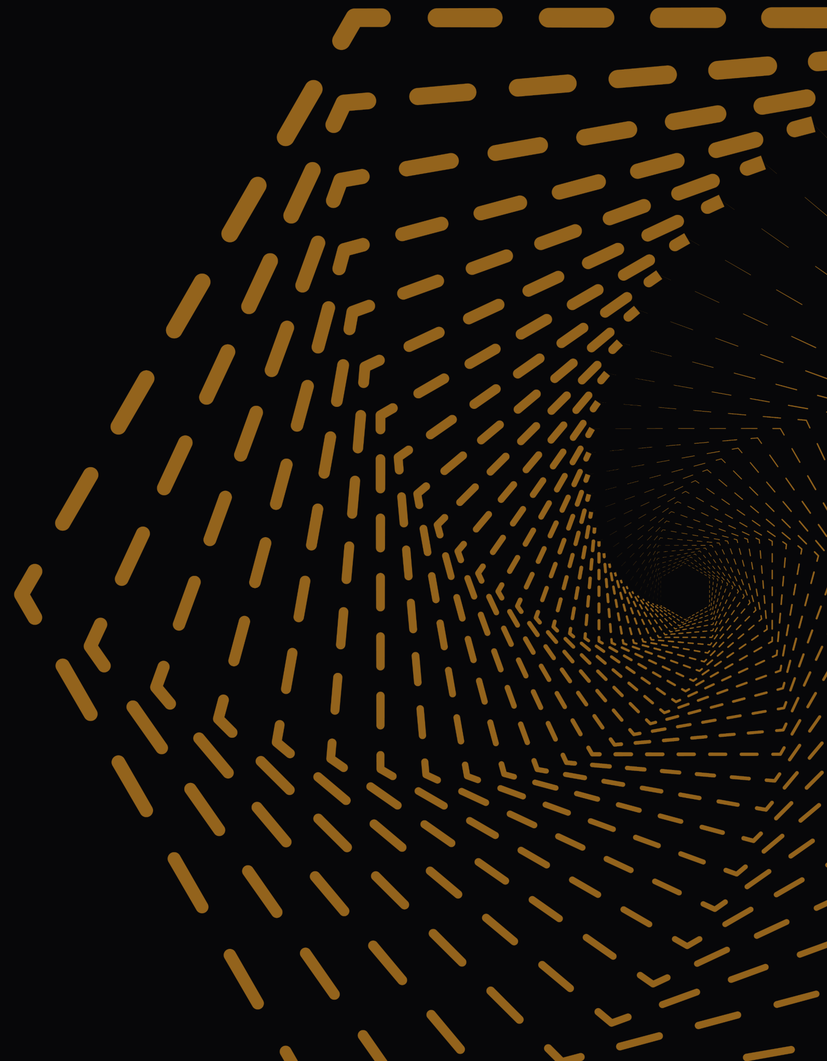
Balun.Courses

ПРОВЕРЬ ЗАПИСЬ



ПРАВИЛА ЗАНЯТИЯ

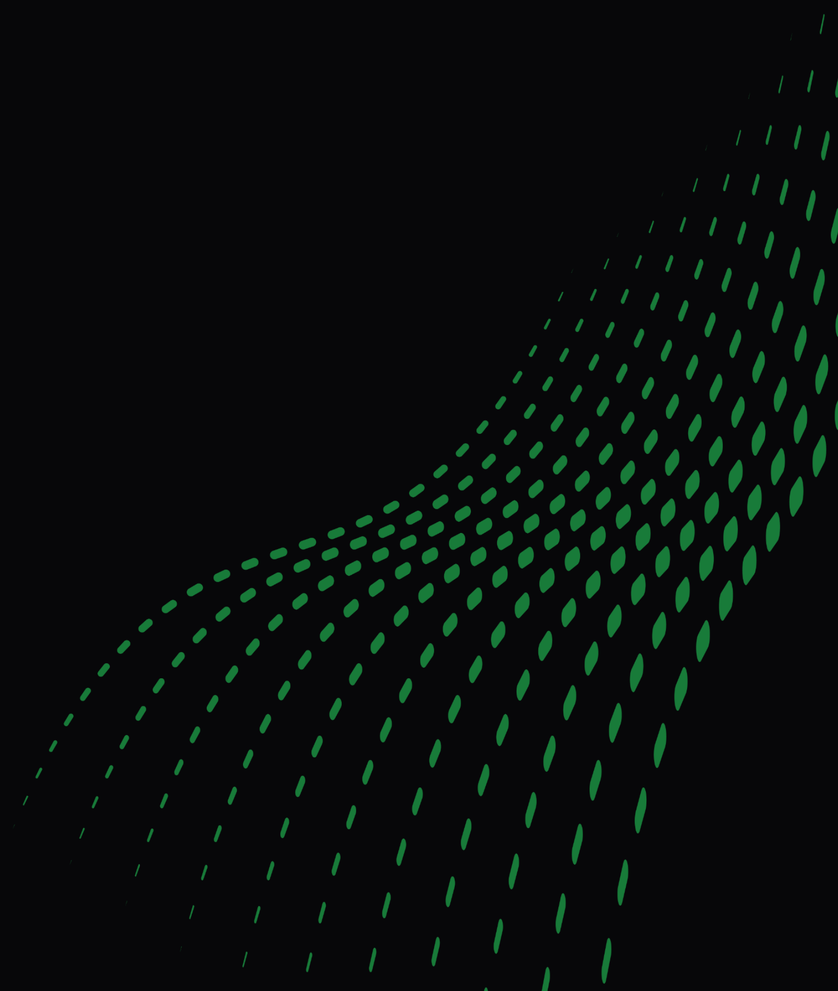
1. вопросы в чате можно задавать **в любое время**
2. вопросы голосом задаем **по поднятой руке в Zoom**
3. ответы на вопросы будут **в запланированных местах**



Balun.Courses

ПЛАН ЛЕКЦИИ

1. Мотивация изучения ассемблера
2. Базовые концепции ассемблера
3. Дизайн ассемблера и компиляции Go
4. Как написать программу на ассемблере Go
5. Отличительные черты ассемблера Go
6. Примеры программ на ассемблере Go
7. SIMD intrinsic functions, почему их нет в Go
8. Примеры программ на Go с использованием SIMD



МОТИВАЦИЯ

Screenshot

```go
func SliceContainsV0(s []uint8, target uint8) bool {  2 usages  Igor Walther
    return slices.Contains(s, target)
}

func SliceContainsV1(s []uint8, target uint8) bool  2 usages  Igor Walther
```

```
go test -bench=. -benchmem -cpu=1
goos: darwin
goarch: arm64
pkg: asm/simd/slice_contains
BenchmarkSliceContains/SliceContainsV1_(SIMD)          42250          26825 ns/op               0 B/op          0 allocs/op
BenchmarkSliceContains/SliceContainsV0                  3802         313418 ns/op               0 B/op          0 allocs/op
PASS
ok      asm/simd/slice_contains 2.878s
```

МОТИВАЦИЯ

Screenshot

```go
func vectorAdditionV0(first, second, dst []uint8) {  2 usages  Igor Walther
    for i := 0; i < len(first); i++ {
        dst[i] = first[i] + second[i]
    }
}

func vectorAdditionV1(first, second, dst []uint8)  2 usages  Igor Walther
```

```
go test -bench=. -benchmem -cpu=1
goos: darwin
goarch: arm64
pkg: asm/simd/vector_addition
BenchmarkAdd/vectorAdditionV1_(SIMD)              45706          27264 ns/op               0 B/op          0 allocs/op
BenchmarkAdd/vectorAdditionV0                      3303         391758 ns/op               0 B/op          0 allocs/op
PASS
ok      asm/simd/vector_addition        3.110s
```

МОТИВАЦИЯ

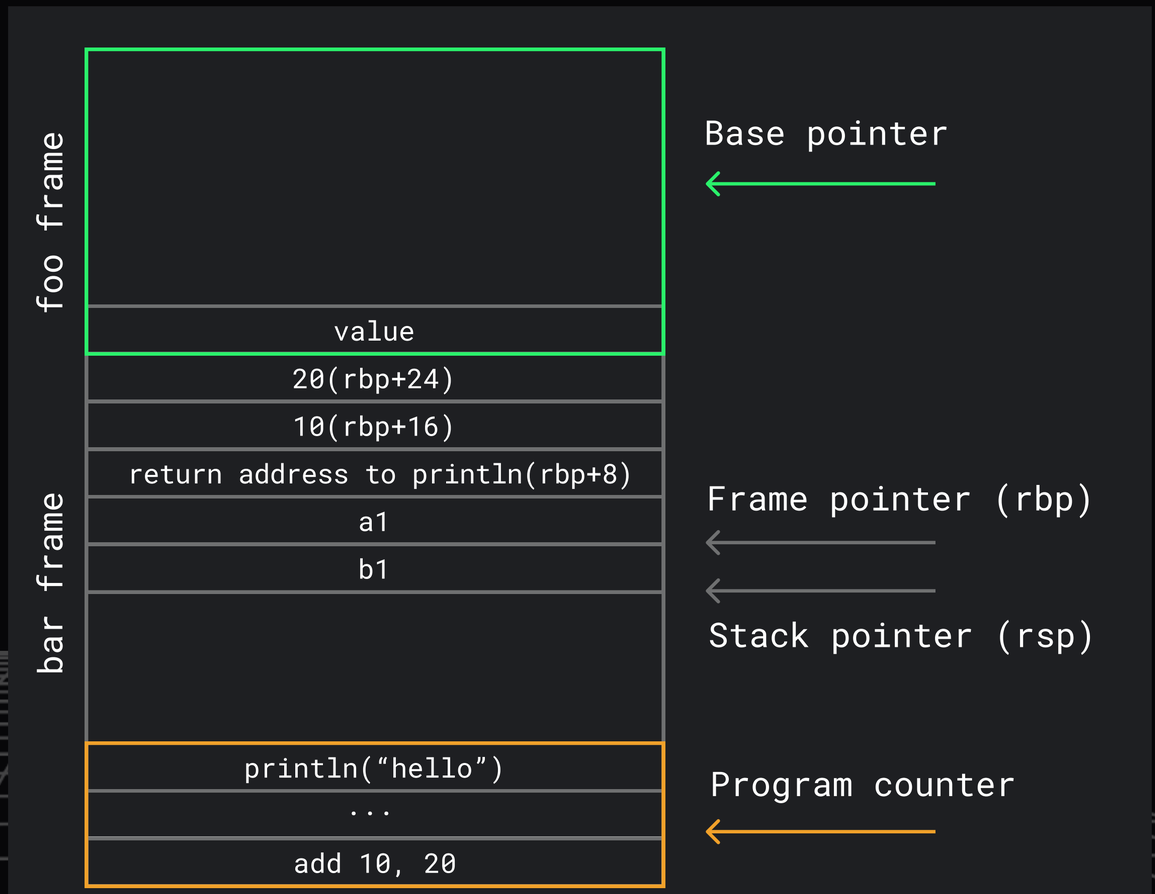
> Also, perhaps most important:
> it is how we talk about the machine.
> Knowing assembly, even a little, means
> understanding computers better

© GopherCon 2016: Rob Pike

АБСТРАКТНЫЙ ПРИМЕР X86

Screenshot

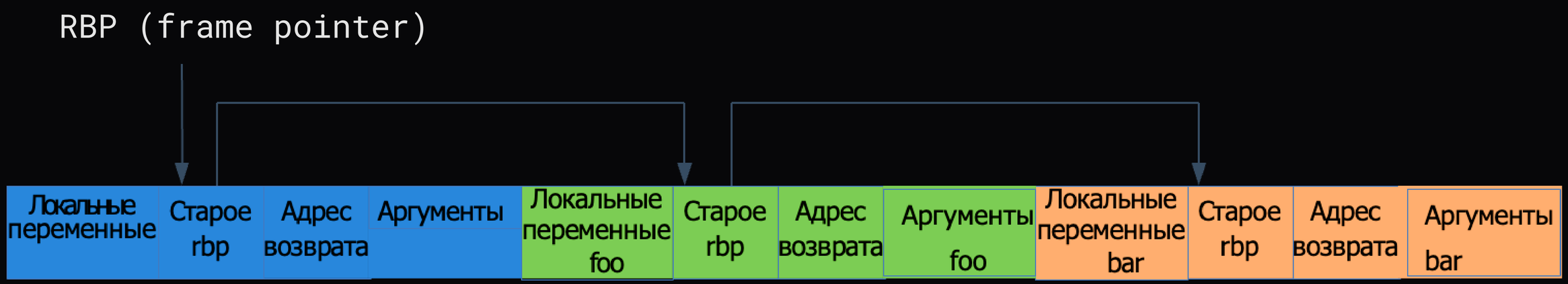
```
func bar() {  no usages
    value := 2
    _ = value
    foo( a: 10,  b: 20)
    println( args...: "hello")
}

func foo(a, b int64) int64 {  1 usage
    a1 := a + 1
    b1 := b + 1

    return a1 + b1
}
```



БАЗОВЫЕ КОНЦЕПЦИИ

Можно ли без frame pointer’а?

Как раскручивать стек?



БАЗОВЫЕ КОНЦЕПЦИИ

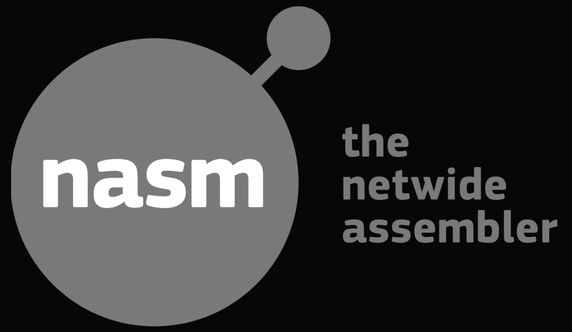
Ассемблерные инструкции - мнемоники для машинных инструкций

Screenshot

```
<unlinkable>.main<1> STEXT size=80 args=0x0 locals=0x18 funcid=0x0 align=0x0
        0x0000 00000 (/Users/igorwalther/GolandProjects/performance_optimizations_go/main.go:3) TEXT    <unlinkable>.main(SB), ABIInternal, $32-0
        0x0000 00000 (/Users/igorwalther/GolandProjects/performance_optimizations_go/main.go:3) MOVD    16(g), R16
        0x0004 00004 (/Users/igorwalther/GolandProjects/performance_optimizations_go/main.go:3) PCDATA  $0, $-2
        0x0004 00004 (/Users/igorwalther/GolandProjects/performance_optimizations_go/main.go:3) CMP     R16, RSP
        0x0008 00008 (/Users/igorwalther/GolandProjects/performance_optimizations_go/main.go:3) BLS     60
        0x000c 00012 (/Users/igorwalther/GolandProjects/performance_optimizations_go/main.go:3) PCDATA  $0, $-1
        0x000c 00012 (/Users/igorwalther/GolandProjects/performance_optimizations_go/main.go:3) MOVD.W  R30, -32(RSP)
        0x0010 00016 (/Users/igorwalther/GolandProjects/performance_optimizations_go/main.go:3) MOVD    R29, -8(RSP)
        0x0014 00020 (/Users/igorwalther/GolandProjects/performance_optimizations_go/main.go:3) SUB     $8, RSP, R29
        0x0018 00024 (/Users/igorwalther/GolandProjects/performance_optimizations_go/main.go:3) FUNCDATA        $0, gclocals·g2BeySu+wFnoycgXfElmcg==(SB)
        0x0018 00024 (/Users/igorwalther/GolandProjects/performance_optimizations_go/main.go:3) FUNCDATA        $1, gclocals·g2BeySu+wFnoycgXfElmcg==(SB)
        0x0018 00024 (/Users/igorwalther/GolandProjects/performance_optimizations_go/main.go:4) PCDATA  $1, $0
        0x0018 00024 (/Users/igorwalther/GolandProjects/performance_optimizations_go/main.go:4) CALL    runtime.printlock(SB)
        0x001c 00028 (/Users/igorwalther/GolandProjects/performance_optimizations_go/main.go:4) MOVD    $go:string."Hello, go course\n"(SB), R0
        0x0024 00036 (/Users/igorwalther/GolandProjects/performance_optimizations_go/main.go:4) MOVD    $17, R1
        0x0028 00040 (/Users/igorwalther/GolandProjects/performance_optimizations_go/main.go:4) CALL    runtime.printstring(SB)
        0x002c 00044 (/Users/igorwalther/GolandProjects/performance_optimizations_go/main.go:4) CALL    runtime.printunlock(SB)
        0x0030 00048 (/Users/igorwalther/GolandProjects/performance_optimizations_go/main.go:5) LDP     -8(RSP), (R29, R30)
        0x0034 00052 (/Users/igorwalther/GolandProjects/performance_optimizations_go/main.go:5) ADD     $32, RSP
        0x0038 00056 (/Users/igorwalther/GolandProjects/performance_optimizations_go/main.go:5) RET     (R30)
        0x003c 00060 (/Users/igorwalther/GolandProjects/performance_optimizations_go/main.go:5) NOP
        0x003c 00060 (/Users/igorwalther/GolandProjects/performance_optimizations_go/main.go:3) PCDATA  $1, $-1
        0x003c 00060 (/Users/igorwalther/GolandProjects/performance_optimizations_go/main.go:3) PCDATA  $0, $-2
        0x003c 00060 (/Users/igorwalther/GolandProjects/performance_optimizations_go/main.go:3) MOVD    R30, R3
        0x0040 00064 (/Users/igorwalther/GolandProjects/performance_optimizations_go/main.go:3) CALL    runtime.morestack_noctxt(SB)
        0x0044 00068 (/Users/igorwalther/GolandProjects/performance_optimizations_go/main.go:3) PCDATA  $0, $-1
        0x0044 00068 (/Users/igorwalther/GolandProjects/performance_optimizations_go/main.go:3) JMP     0
        0x0000 90 0b 40 f9 ff 63 30 eb a9 01 00 54 fe 0f 1e f8  ..@..c0....T....
        0x0010 fd 83 1f f8 fd 23 00 d1 00 00 00 94 00 00 00 90  .....#..........
        0x0020 00 00 00 91 21 02 80 d2 00 00 00 94 00 00 00 94  ....!...........
        0x0030 fd fb 7f a9 ff 83 00 91 c0 03 5f d6 e3 03 1e aa  .........._.....
        0x0040 00 00 00 94 ef ff ff 17 00 00 00 00 00 00 00 00  ................
```

```go
package performance_optimizations_go

func main() {  no usages
    println( args...: "Hello, go course")
}
```

БАЗОВЫЕ КОНЦЕПЦИИ



NASM

rax	mov	rdi	mov rax, rdi
			mov rax, 5
			mov rax, BYTE [rdi]

rax	add	rdi	add rax, rdi
			add rax, 5
			add rax, [rdi]

rax	sub	rdi	sub rax, rdi
			sub rax, 5
			sub rax, WORD [rdi]

БАЗОВЫЕ КОНЦЕПЦИИ

NASM

rax	mov	rdi	`mov rax, rdi`
			`mov rax, 5`
			`mov rax, DWORD [rdi]`

rax	add	rdi	`add rax, rdi`
			`add rax, 5`
			`add rax, QWORD [rdi]`

rax	sub	rdi	`sub rax, rdi`
			`sub rax, 5`
			`sub rax, [rdi]`

БАЗОВЫЕ КОНЦЕПЦИИ

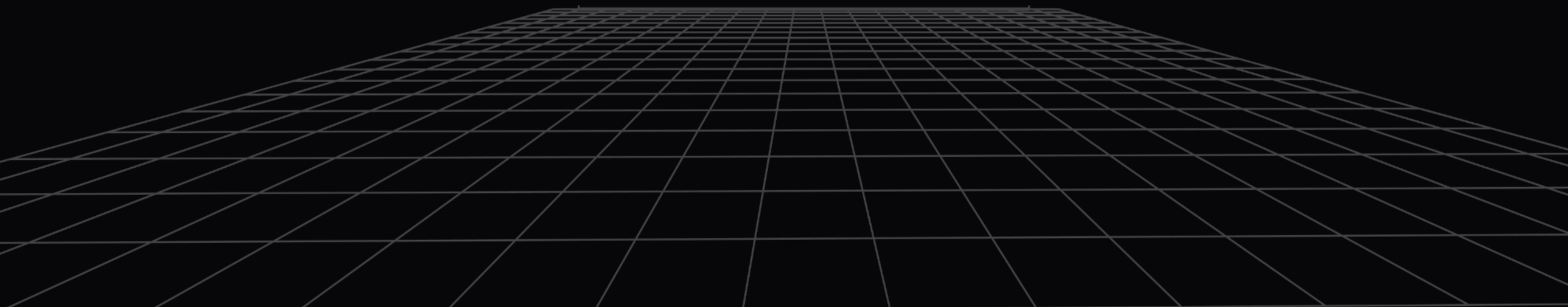
NASM


```
xor rax, rdi
xor rax, 5
xor rax, [rdi]
```

jz - jump if zero

Пример инструкции управления после операции xor.
В случае zero flag подвинем PC на другую инструкцию

DEMO



НА КАКОМ ЭТАПЕ КОМПИЛЯТОР GO ГЕНЕРИРУЕТ АССЕМБЛЕРНЫЙ КОД?

КОМПИЛЯЦИЯ

https://github.com/golang/go/blob/master/src/cmd/compile/internal/syntax/parser.go

1. Парсинг исходного кода

```
type parser struct {
    file  *PosBase
    errh  ErrorHandler
    mode  Mode
    pragh PragmaHandler
    scanner

    base      *PosBase // current position base
    first     error    // first error encountered
    errcnt    int      // number of errors encountered
    pragma    Pragma   // pragmas
    goVersion string   // Go version from //go:build line

    top    bool   // in top of file (before package clause)
    fnest  int    // function nesting level (for error handling)
    xnest  int    // expression nesting level (for complit ambiguity resolution)
    indent []byte // tracing support
}
```

sources

```
   /Users/igorwalther/GolandProjects/p
3  func main() {
4      println("Go ASM digest")
5  }
```

КОМПИЛЯЦИЯ

https://github.com/golang/go/tree/master/src/cmd/compile/internal/types2

2. Семантический анализ (AST, type checking)

```
    SwitchStmt struct {
        Init   SimpleStmt
        Tag    Expr // incl. *TypeSwitchGuard
        Body   []*CaseClause
        Rbrace Pos
        stmt
    }

    SelectStmt struct {
        Body   []*CommClause
        Rbrace Pos
        stmt
    }
)
```

AST

```
buildssa-body
. BLOCK tc(1) # main.go:4:9
. BLOCK-List
. . CALLFUNC Walked tc(1) # main.go:4:9
. . CALLFUNC-Fun
. . . NAME-runtime.printlock Class:PFUNC Offset:0 Used F
. . CALLFUNC Walked tc(1) # main.go:4:9
. . CALLFUNC-Fun
. . . NAME-runtime.printstring Class:PFUNC Offset:0 Used
. . CALLFUNC-Args
. . . LITERAL-"Go ASM digest\n" string tc(1) # main.go:4
. . CALLFUNC Walked tc(1) # main.go:4:9
. . CALLFUNC-Fun
. . . NAME-runtime.printunlock Class:PFUNC Offset:0 Used
```

КОМПИЛЯЦИЯ

https://github.com/golang/go/tree/master/src/cmd/compile/internal/ir

3. Compiler intermediate representation (noding)

```go
// A Node is the abstract interface to an IR node.
type Node interface {
    // Formatting
    Format(s fmt.State, verb rune)

    // Source position.
    Pos() src.XPos
    SetPos(x src.XPos)

    // For making copies. For Copy and SepCopy.
    copy() Node

    doChildren(func(Node) bool) bool
    editChildren(func(Node) Node)
    editChildrenWithHidden(func(Node) Node)
```

КОМПИЛЯЦИЯ

https://github.com/golang/go/tree/master/src/cmd/compile/internal/ssagen

4. Преобразование IR в Static Single Assignment form

```go
// buildssa builds an SSA function for fn.
// worker indicates which of the backend workers is doing the processing.
func buildssa(fn *ir.Func, worker int) *ssa.Func {
    name := ir.FuncName(fn)

    abiSelf := abiForFunc(fn, ssaConfig.ABI0, ssaConfig.ABI1)

    printssa := false
    // match either a simple name e.g. "(*Reader).Reset", package.name e.g.
    // optionally allows an ABI suffix specification in the GOSSAHASH, e.g.
    if strings.Contains(ssaDump, name) { // in all the cases the function n
        nameOptABI := name
```

```
b1:
  v2 (?) = SB <uintptr> : SB
  v1 (?) = InitMem <mem>
  v4 (+4) = CALLstatic <mem>
     {AuxCall{runtime.printlock}} v1
  v5 (4) = SelectN <mem> [0] v4
  v14 (4) = MOVDaddr <*uint8> {go:string."Go ASM
     digest\n"} v2 : R0
  v13 (4) = MOVDconst <int> [14] : R1
  v7 (4) = CALLstatic <mem>
     {AuxCall{runtime.printstring}} [16] v14 v13 v5 :
     <>
  v8 (4) = SelectN <mem> [0] v7
  v9 (4) = CALLstatic <mem>
     {AuxCall{runtime.printunlock}} v8
  v10 (4) = SelectN <mem> [0] v9
  v11 (+5) = MakeResult <mem> v10
Ret v11 (5)
```

КОМПИЛЯЦИЯ

https://github.com/golang/go/blob/master/src/cmd/compile/internal/arm64/ssa.go

5. Генерация ассемблера и машинного кода

```
// storeByType returns the store instruction of the given type.
func storeByType(t *types.Type) obj.As {
    if t.IsFloat() {...} else {
        switch t.Size() {
        case 1: return arm64.AMOVB
        case 2: return arm64.AMOVH
        case 4: return arm64.AMOVW
        case 8: return arm64.AMOVD
        }
    }
    panic( v: "bad store type")
}
```

КОМПИЛЯЦИЯ

5. Генерация ассемблера и машинного кода

```
# /Users/igorwalther/GolandProjects/performance_optimizations_go/main.go
00000 (3) TEXT main.main(SB), ABIInternal
00001 (3) FUNCDATA $0, gclocals·g2BeySu+wFnoycgXfElmcg==(SB)
00002 (3) FUNCDATA $1, gclocals·g2BeySu+wFnoycgXfElmcg==(SB)
00003 (+4) PCDATA $1, $0
00004 (+4) CALL runtime.printlock(SB)
00005 (4) MOVD $go:string."Go ASM digest\n"(SB), R0
00006 (4) MOVD $14, R1
00007 (4) CALL runtime.printstring(SB)
00008 (4) CALL runtime.printunlock(SB)
00009 (5) RET
00010 (?) END
```

```
GOSSAFUNC=main go tool compile main.go
```

ГДЕ ЖИВЕТ АССЕМБЛЕР GO?

Lexemes → Abstract syntax tree → Intermediate representation (SSA) → Go assembler → Machine code

```
# /Users/igorwalther/GolandProjects/performance_optimizations_go/main.go
00000 (3) TEXT main.main(SB), ABIInternal
00001 (3) FUNCDATA $0, gclocals·g2BeySu+wFnoycgXfElmcg==(SB)
00002 (3) FUNCDATA $1, gclocals·g2BeySu+wFnoycgXfElmcg==(SB)
00003 (+4) PCDATA $1, $0
00004 (+4) CALL runtime.printlock(SB)
00005 (4) MOVD $go:string."Go ASM digest\n"(SB), R0
00006 (4) MOVD $14, R1
00007 (4) CALL runtime.printstring(SB)
00008 (4) CALL runtime.printunlock(SB)
00009 (5) RET
00010 (?) END
```

```
0x0000 90 0b 40 f9 ff 63 30 eb a9 01 00 54 fe 0f 1e f8  .
0x0010 fd 83 1f f8 fd 23 00 d1 00 00 00 94 00 00 00 90  .
0x0020 00 00 00 91 e1 0b 7f b2 00 00 00 94 00 00 00 94  .
0x0030 fd fb 7f a9 ff 83 00 91 c0 03 5f d6 e3 03 1e aa  .
0x0040 00 00 00 94 ef ff ff 17 00 00 00 00 00 00 00 00  .
```

НЕКОТОРЫЕ ИНСТРУКЦИИ И РЕГИСТРЫ ЗАВИСЯТ ОТ ПЛАТФОРМЫ

```
MOVQ x_base+0(FP), AX // ptr
MOVQ x_len+8(FP), DX // len
```

amd64

```
MOVD x_base+0(FP), R0 // ptr
MOVD x_len+8(FP), R1 // len
```

arm64

НЕКОТОРЫЕ ИНСТРУКЦИИ И РЕГИСТРЫ ЗАВИСЯТ ОТ ПЛАТФОРМЫ

```
const (
    REG_AL = obj.RBaseAMD64 + iota
    REG_CL
    REG_DL
    REG_BL
    REG_SPB
    REG_BPB
    REG_SIB
    REG_DIB
    REG_R8B
    REG_R9B
    REG_R10B
    REG_R11B
    REG_R12B
    REG_R13B
    REG_R14B
    REG_R15B
```

amd64

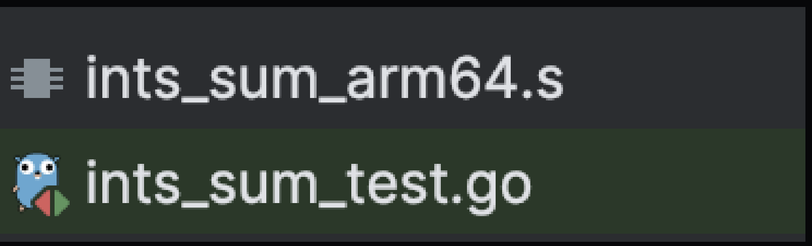
```
const (
    // integer
    REG_R0 = obj.RBaseARM64 + iota
    REG_R1
    REG_R2
    REG_R3
    REG_R4
    REG_R5
    REG_R6
    REG_R7
    REG_R8
    REG_R9
    REG_R10
    REG_R11
    REG_R12
    REG_R13
    REG_R14
    REG_R15
```

arm64

КАК НАПИСАТЬ ПРОГРАММУ НА АССЕМБЛЕРЕ

```
func IntSum(a, b int64) int64  1 usage  Igor Walther
```

```
#include "textflag.h"

// func IntSum(a, b int64) int64
TEXT ·IntSum(SB), NOSPLIT, $0
    LDP slice_base+0(FP), (R0, R1)
    ADD R0, R1
    MOVD R1, ret+16(FP)
    RET
```

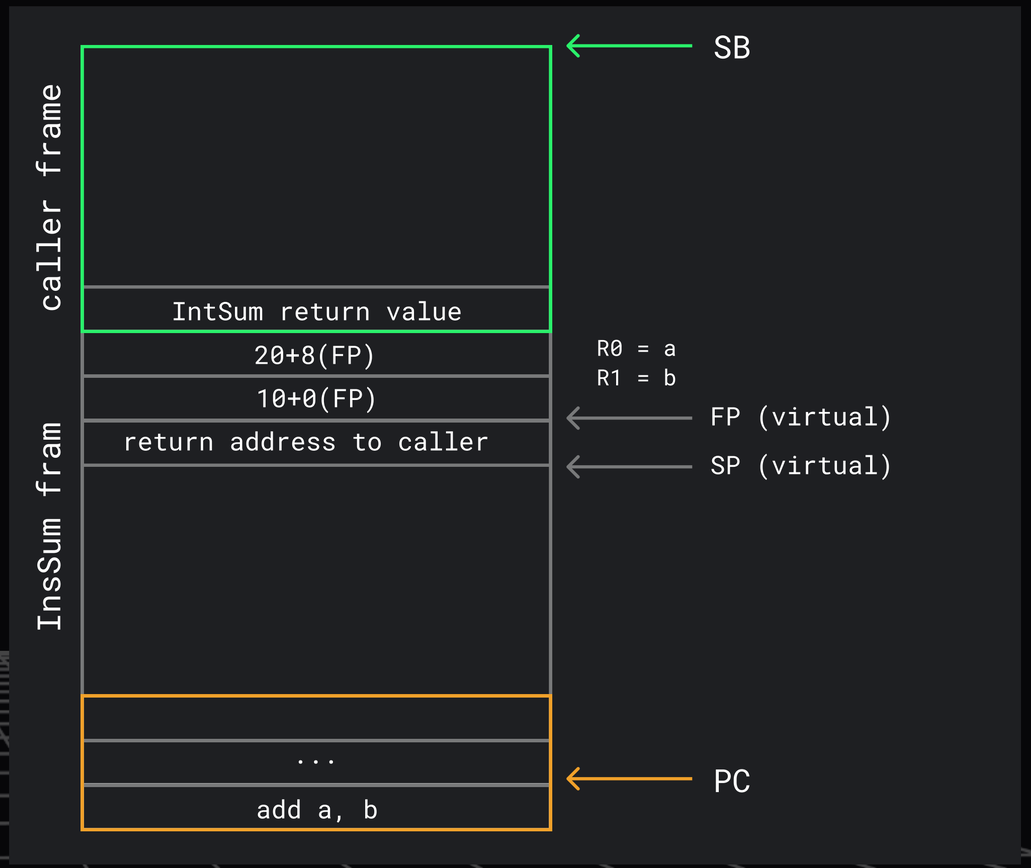


textflag.h - стандартные директивы
· / - разделители, (math/rand·Int)
FP - указатель фрейма (for local)
SB - static base указатель (for global)

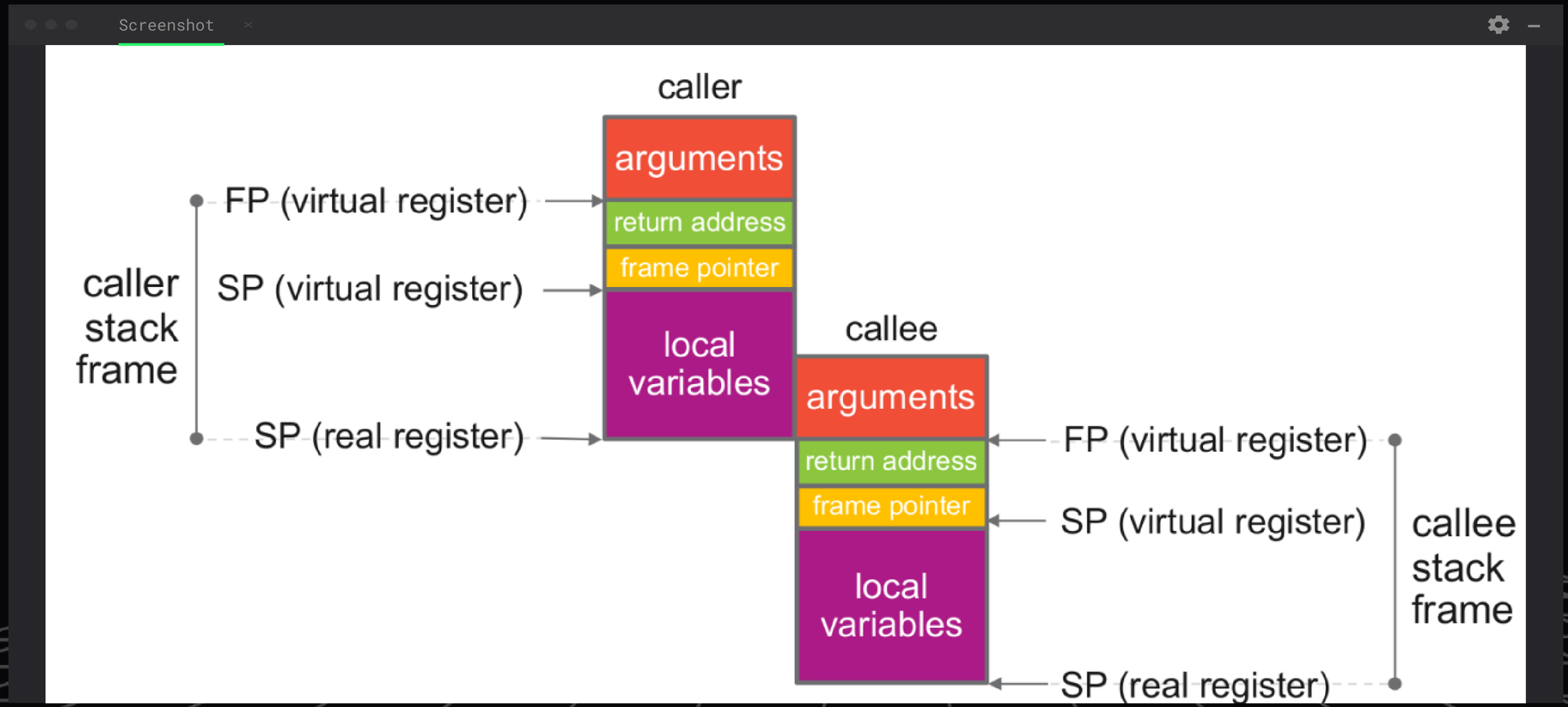
НА САМОМ ДЕЛЕ СЛОЖНЕЕ

Screenshot

```
func IntSum(a, b int64) int64  1 usage  Igor Walther

#include "textflag.h"

// func IntSum(a, b int64) int64
TEXT ·IntSum(SB), NOSPLIT, $0
    LDP slice_base+0(FP), (R0, R1)
    ADD R0, R1
    MOVD R1, ret+16(FP)
    RET
```



НА САМОМ ДЕЛЕ СЛОЖНЕЕ



ОТЛИЧИТЕЛЬНЫЕ ЧЕРТЫ АССЕМБЛЕРА GO

1. В ассемблере Go изменен порядок аргументов для большинства инструкций

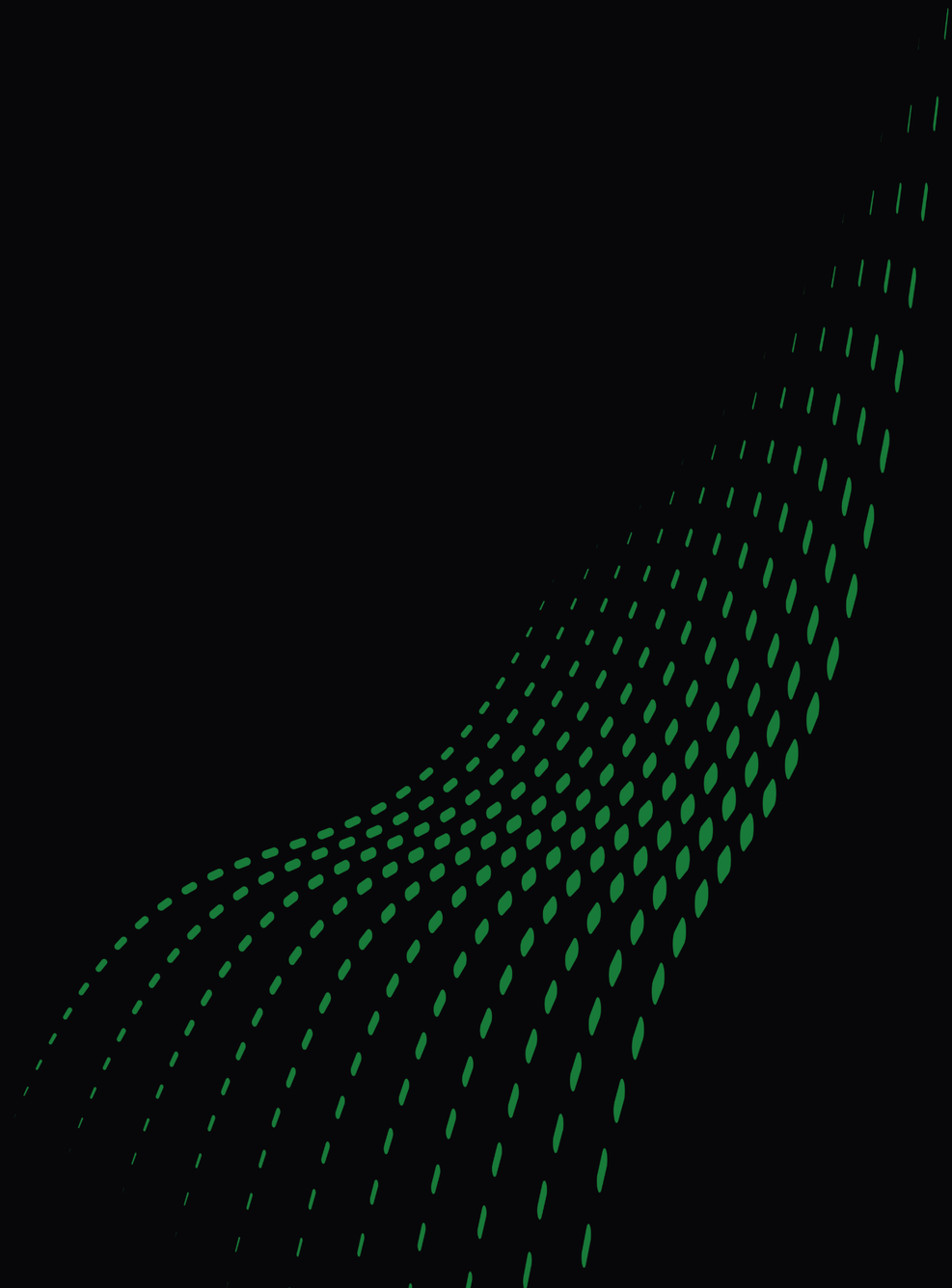
   MOV src, dst - go amd64 (left-to-right assignment order)
   MOV dst, src - amd64

2. В ассемблере Go инструкция NOP не преобразуется в машинный код

ОТЛИЧИТЕЛЬНЫЕ ЧЕРТЫ АССЕМБЛЕРА GO

3. В ассемблере Go для большинства floating-point и SIMD инструкций добавлен префикс V (VLD1, VST1, VADD, VFMLA)

4. Некоторые инструкции на различных платформах не поддержаны. Иногда приходится декларировать их как последовательность байт, используя директивы BYTE и WORD (WORD $0x6e20dde2 - VFMUL V0, V15, V2)



ОТЛИЧИТЕЛЬНЫЕ ЧЕРТЫ АССЕМБЛЕРА GO

https://github.com/golang/go/blob/master/src/cmd/internal/obj/link.go#L704

5. Internal и stable ABI (calling convention)

```
// ABI is the calling convention of a text symbol.
type ABI uint8

const (
    // ABI0 is the stable stack-based ABI. It's important that the
    // value of this is "0": we can't distinguish between
    // references to data and ABI0 text symbols in assembly code,
    // and hence this doesn't distinguish between symbols without
    // an ABI and text symbols with ABI0.
    ABI0 = 0 ABI = iota

    // ABIInternal is the internal ABI that may change between Go
    // versions. All Go functions use the internal ABI and the
    // compiler generates wrappers for calls to and from other
    // ABIs.
    ABIInternal = 1
```

ОТЛИЧИТЕЛЬНЫЕ ЧЕРТЫ АССЕМБЛЕРА GO

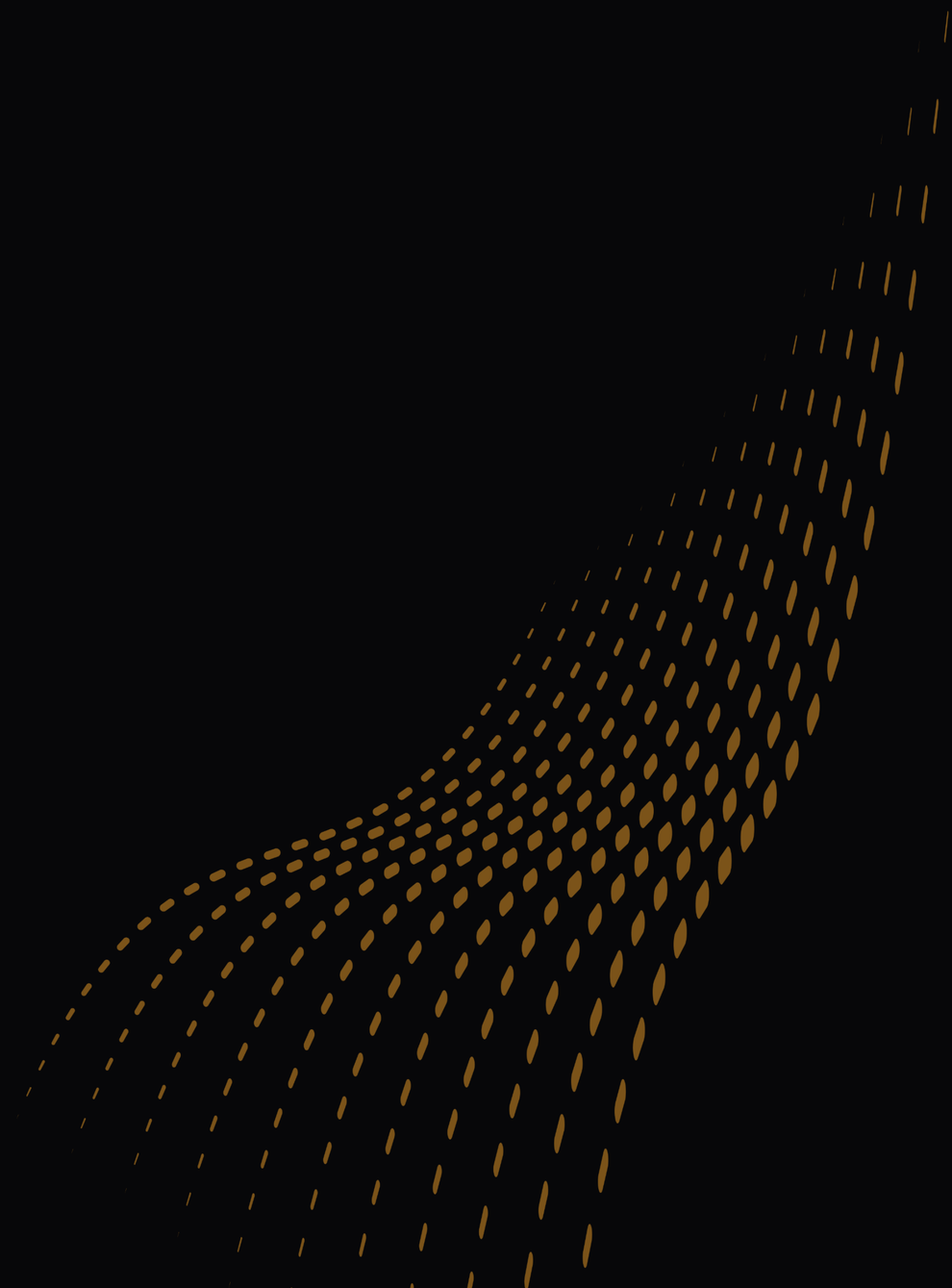
https://github.com/golang/go/blob/master/src/cmd/internal/obj/link.go#L704

6. Alignment

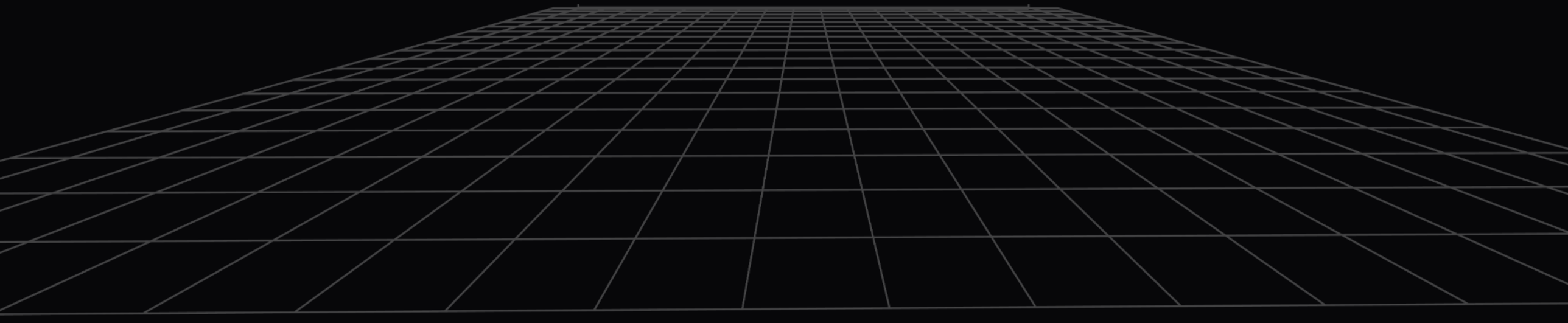
```
PCALIGN $16
MOVD $2, R0          // This instruction is aligned with 16 bytes.
PCALIGN $1024
MOVD $3, R1          // This instruction is aligned with 1024 bytes.
```

ПРИМЕРЫ ПРОГРАММ НА АССЕМБЛЕРЕ GO

1. Sum
2. Slice sum
3. Fibonacci
4. Vector operations

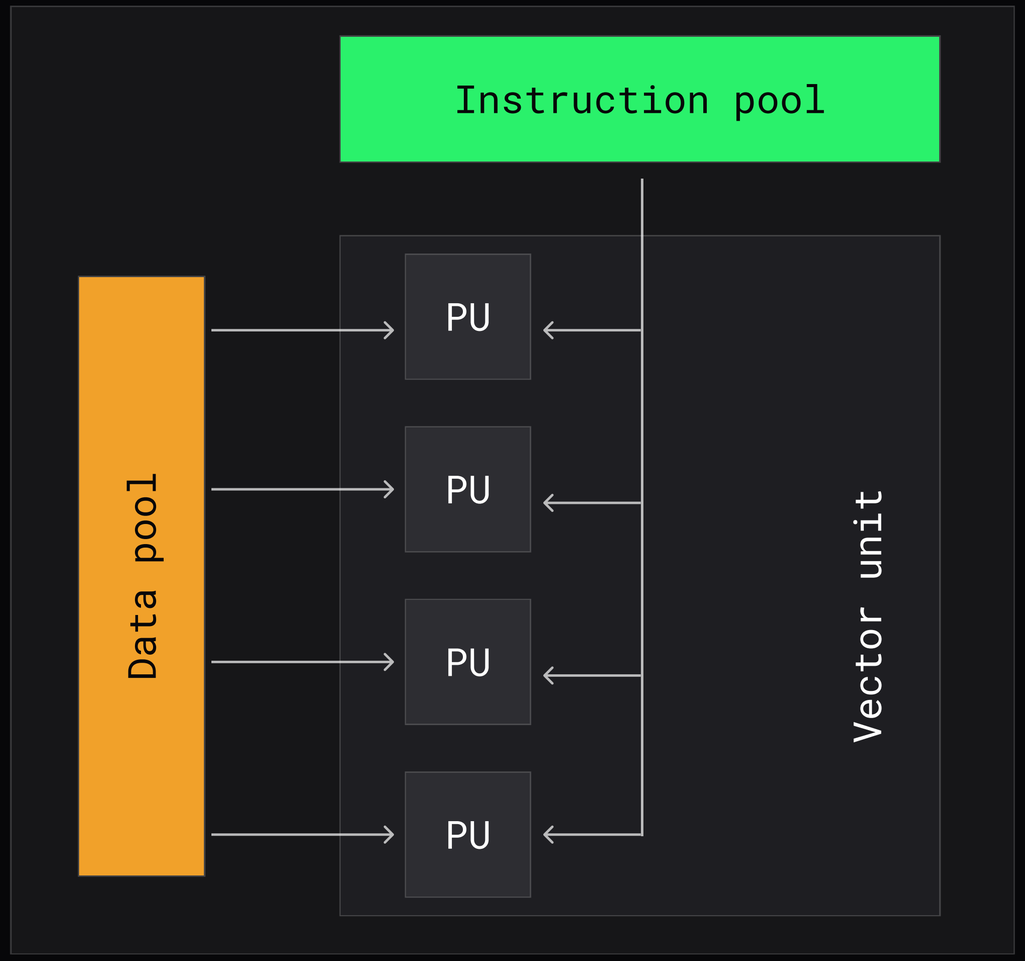


DEMO



SIMD

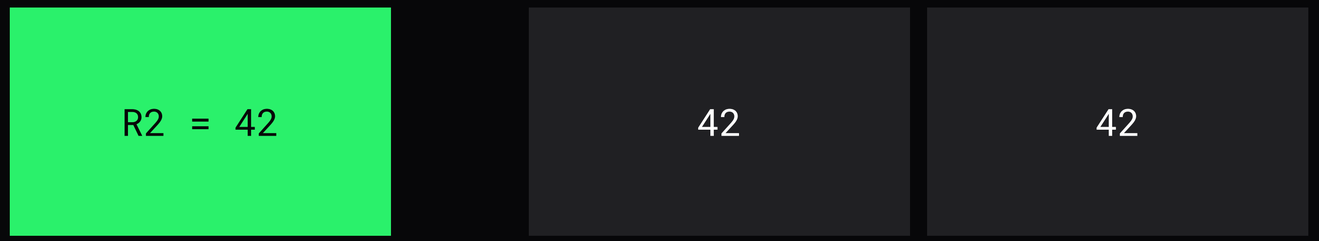
Повышение производительности (SIMD, etc)



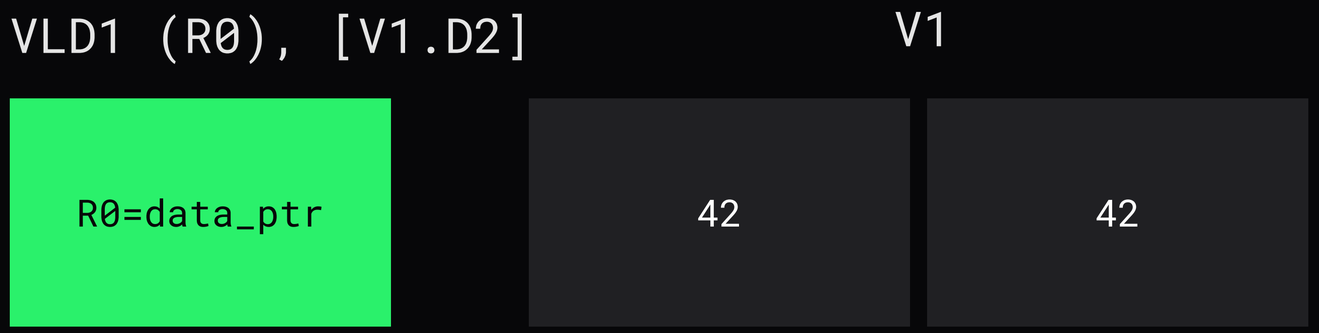

```
VADD V1.D2, V2.D2, V3.D2
```

ВЕКТОРНЫЕ ИНСТРУКЦИИ

VDUP R2, V1.D2



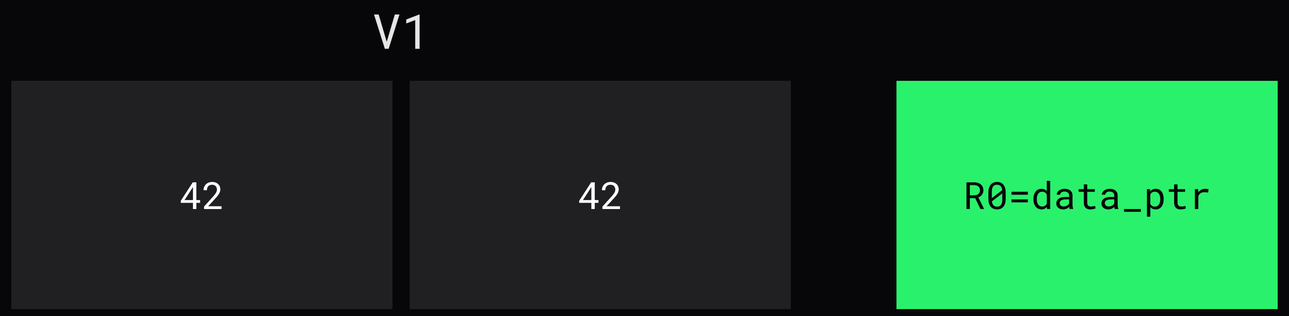
VLD1 (R0), [V1.D2]



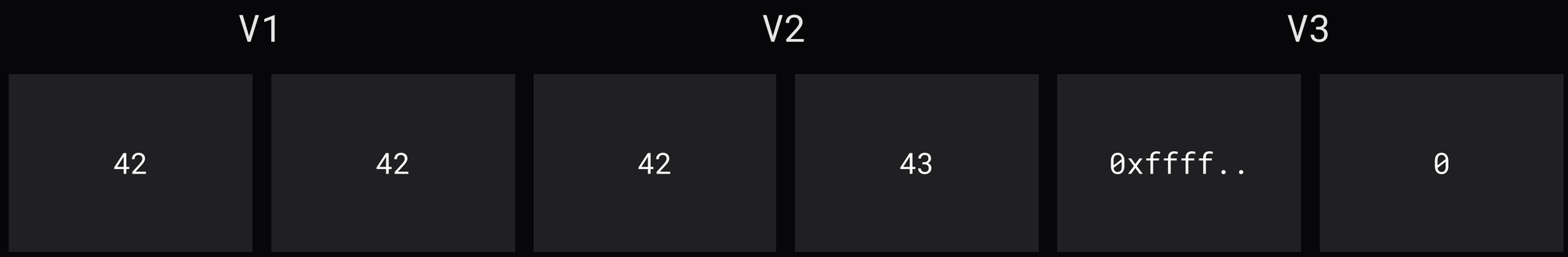
ВЕКТОРНЫЕ ИНСТРУКЦИИ

Повышение производительности (SIMD, etc)

VST1 [V1.D2], (R0)



VCMEQ V1.D2, V2.D2, V3.D2



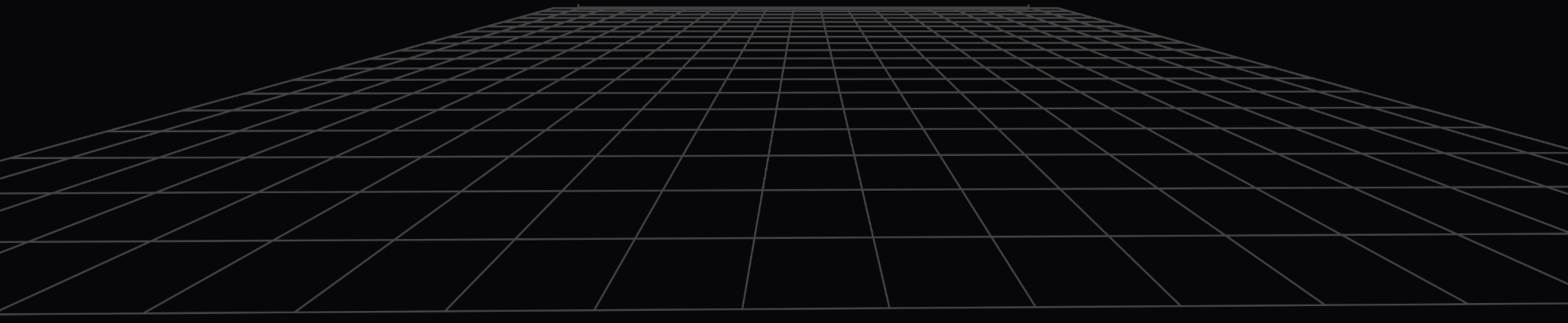
SIMD INTRINSIC FUNCTIONS, ПОЧЕМУ ИХ НЕТ В GO

1. Сложность поддержки для большинства архитектур в текущей грамматике SSA
2. “It would be fun, but other stuff is more pressing”

ПРИМЕРЫ ПРОГРАММЫ НА АССЕМБЛЕРЕ GO С ИСПОЛЬЗОВАНИЕМ SIMD

DEMO

Примеры программы на ассемблере GO с использованием SIMD



ИТОГИ И ВЫВОДЫ

ЧТО БУДЕТ ДАЛЬШЕ