Why cgo is slow

Filippo Valsorda

```go
package main

// int my_C_function(int a, int b) {
//     return a + b;
// }
import "C"

func main() {
    a, b := C.int(40), C.int(2)
    c := C.my_C_function(a, b)
    println(a, b, c)
}
```

cgo is a FFI

(Foreign Function Interface)

I like FFIs.

- From cgo back to Go @ GopherCon 2016

  https://speakerdeck.com/filosottile/from-cgo-back-to-go-gophercon-2016
- rustgo: Building your own FFI @ GothamGo 2017

  https://speakerdeck.com/filosottile/calling-rust-from-go-without-cgo-at-gothamgo-2017
- Why cgo is slow @ CapitalGo 2018

  Hi!

C function call	2.364 ns
Java FFI	9.01 ns
Rust FFI	2.386 ns
LuaJIT FFI	1.81 ns (!) https://nullprogram.com/blog/2018/05/27/
Node.js FFI	18.33 ns
cgo	75.95 ns

https://github.com/dyu/ffi-overhead

```
name          old time/op  new time/op  delta
CgoCall-4     63.1ns ± 3%  57.1ns ± 0%  -9.43%
```

C function call	2.364 ns
Java FFI	9.01 ns
Rust FFI	2.386 ns
LuaJIT FFI	1.81 ns (!) https://nullprogram.com/blog/2018/05/27/
Node.js FFI	18.33 ns
cgo	68.77 ns

https://github.com/dyu/ffi-overhead

C function call	2.364 ns
Java FFI	9.01 ns
Rust FFI	2.386 ns
LuaJIT FFI	1.81 ns (!) https://nullprogram.com/blog/2018/05/27/
Node.js FFI	18.33 ns
cgo	68.77 ns (29x)

https://github.com/dyu/ffi-overhead

cgo:

- cmd/cgo
- runtime/cgo
- a sprinkle of cmd/link/internal/ld support

cgo:

- cmd/cgo
- runtime/cgo
- a sprinkle of cmd/link/internal/ld support
- not a compiler feature!

cgo:

- cmd/cgo — a code generator
- runtime/cgo
- a sprinkle of cmd/link/internal/ld support
- not a compiler feature!

Reason 1:

calling conventions

C compiler

Go compiler

```
package main

// int my_C_function(int a, int b) {
//     return a + b;
// }
import "C"

func main() {
    a, b := C.int(40), C.int(2)
    c := C.my_C_function(a, b)
    println(a, b, c)
}
```

```
go build -x -work
```

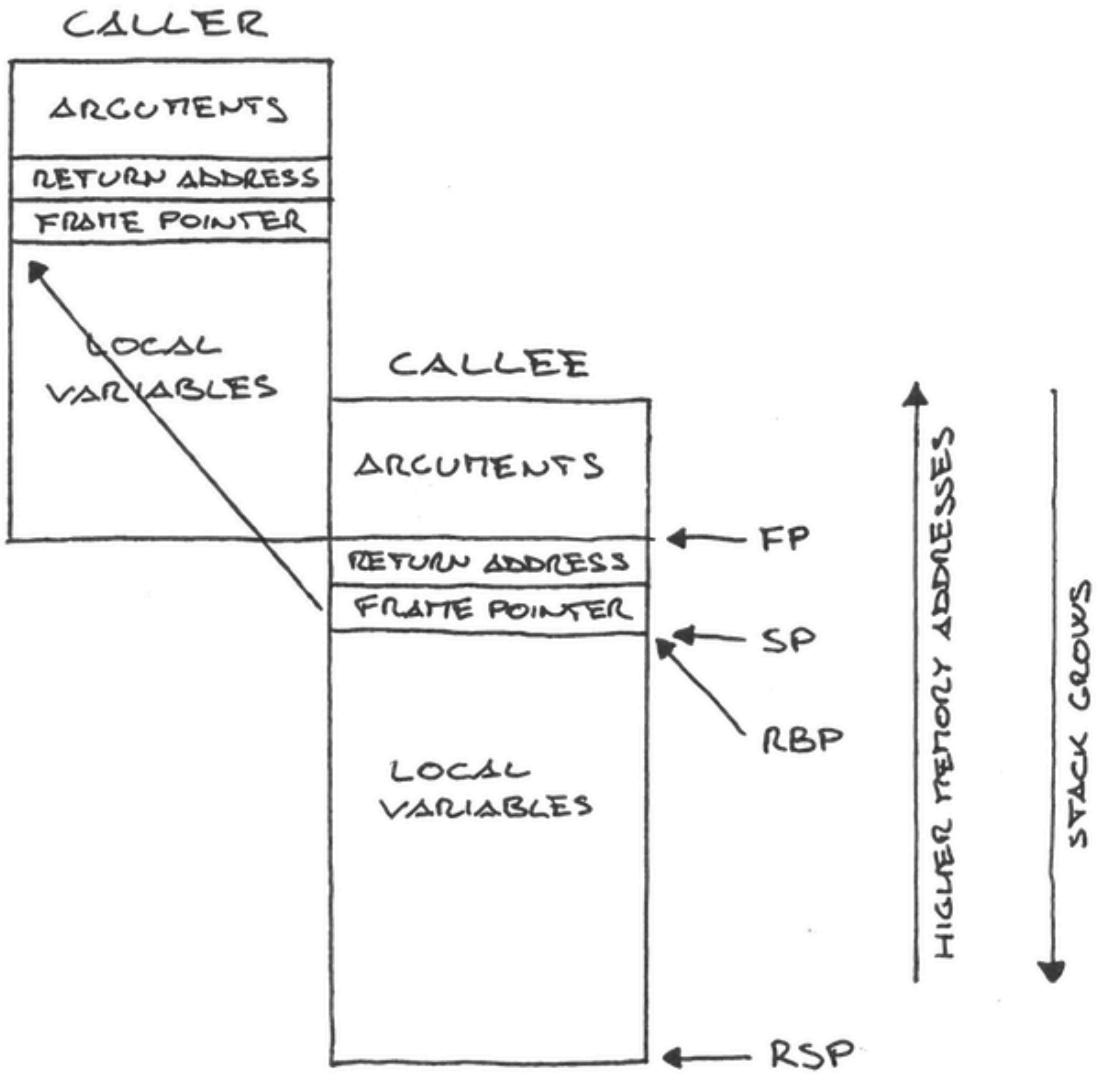
```go
// Created by cgo - DO NOT EDIT

package main

// int my_C_function(int a, int b) {
//     return a + b;
// }
import _ "unsafe"

func main() {
    a, b := _Ctype_int(40), _Ctype_int(2)
    c := (_Cfunc_my_C_function)(a, b)
    println(a, b, c)
}
```

```go
func _Cfunc_my_C_function(p0, p1 _Ctype_int) (r1 _Ctype_int) {
    runtime.cgocall(_cgo_Cfunc_my_C_function,
        uintptr(unsafe.Pointer(&p0)))
    return
}
```



CALLER
ARGUMENTS
RETURN ADDRESS
FRAME POINTER
LOCAL VARIABLES
CALLEE
ARGUMENTS
RETURN ADDRESS
FRAME POINTER
LOCAL VARIABLES
FP
SP
RBP
RSP
HIGHER MEMORY ADDRESSES
STACK GROWS

```
func _Cfunc_my_C_function(p0, p1 _Ctype_int) (r1 _Ctype_int) {
    runtime.cgocall(_cgo_Cfunc_my_C_function,
        uintptr(unsafe.Pointer(&p0)))
    return
}
```

```
src/runtime/cgocall.go
```

```
// func asmcgocall(fn, arg unsafe.Pointer) int32
// Call fn(arg) on the scheduler stack,
// aligned appropriately for the gcc ABI.
// See cgocall.go for more details.
TEXT ·asmcgocall(SB),NOSPLIT,$0-20
    MOVQ    fn+0(FP), AX
    MOVQ    arg+8(FP), BX

    MOVQ    BX, DI      // DI = first argument in AMD64 ABI
    MOVQ    BX, CX      // CX = first argument in Win64
    CALL    AX

    MOVL    AX, ret+16(FP)
    RET
```

```
// func asmcgocall(fn, arg unsafe.Pointer) int32
// Call fn(arg) on the scheduler stack,
// aligned appropriately for the gcc ABI.
// See cgocall.go for more details.
TEXT ·asmcgocall(SB),NOSPLIT,$0-20
    MOVQ    fn+0(FP), AX
    MOVQ    arg+8(FP), BX

    MOVQ    BX, DI          // DI = first argument in AMD64 ABI
    MOVQ    BX, CX          // CX = first argument in Win64
    CALL    AX

    MOVL    AX, ret+16(FP)
    RET
```

```c
void _cgo_Cfunc_my_C_function(void *v)
{
    struct {
        int p0;
        int p1;
        int r;
    } *a = v;
    a->r = my_C_function(a->p0, a->p1);
}
```

Go	`_Cfunc_my_C_function →`	rewritten Go function
ASM	`runtime.[asm]cgocall →`	calling convention trampoline
C	`_cgo_Cfunc_my_C_function →`	arg unpacking
	`my_C_function`	real C function

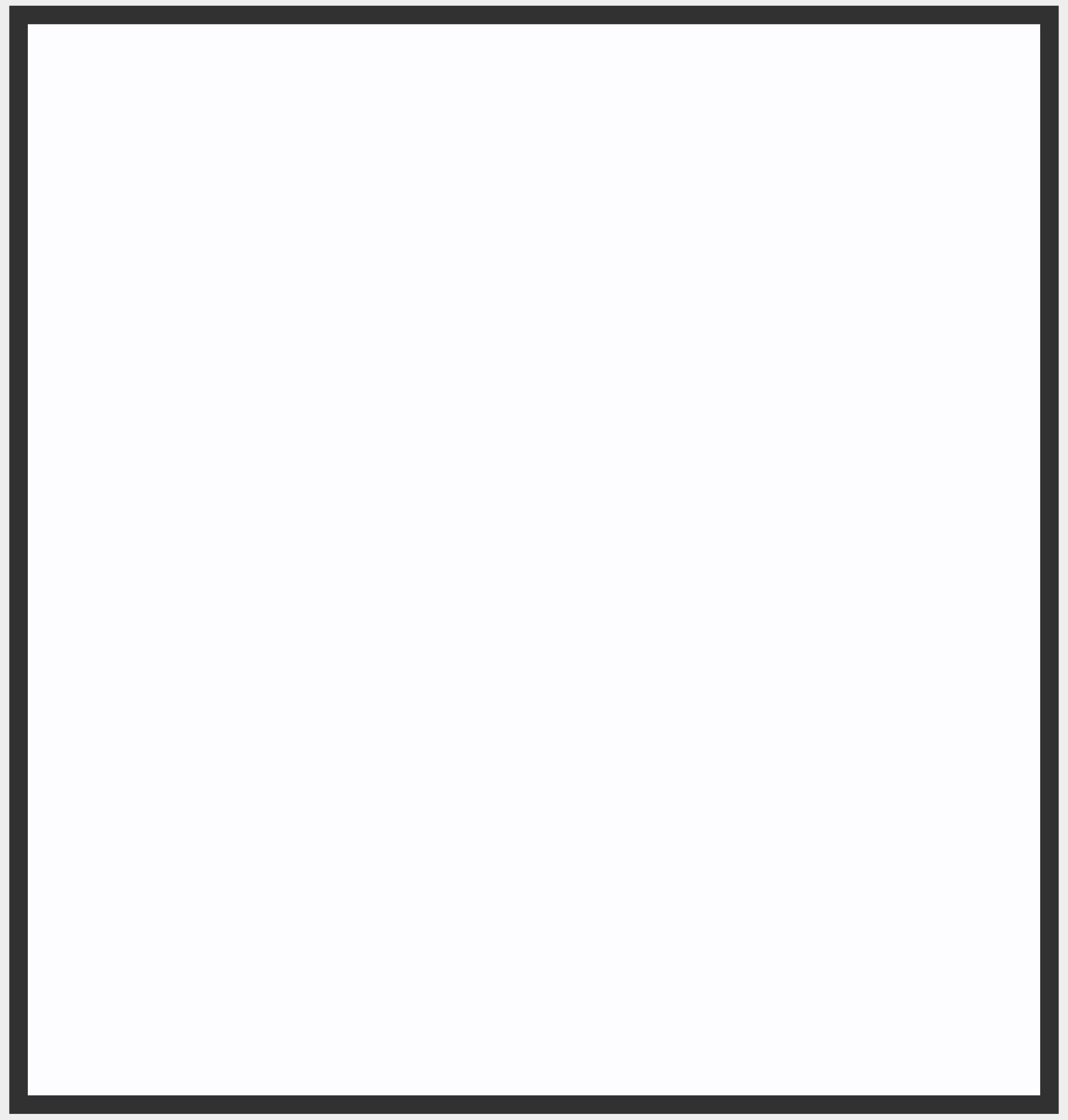
Learn more:

- src/runtime/cgocall.go
- rustgo: Building your own FFI @ GothamGo 2017
  https://speakerdeck.com/filosottile/calling-rust-from-go-without-cgo-at-gothamgo-2017

Reason 2:
small stacks

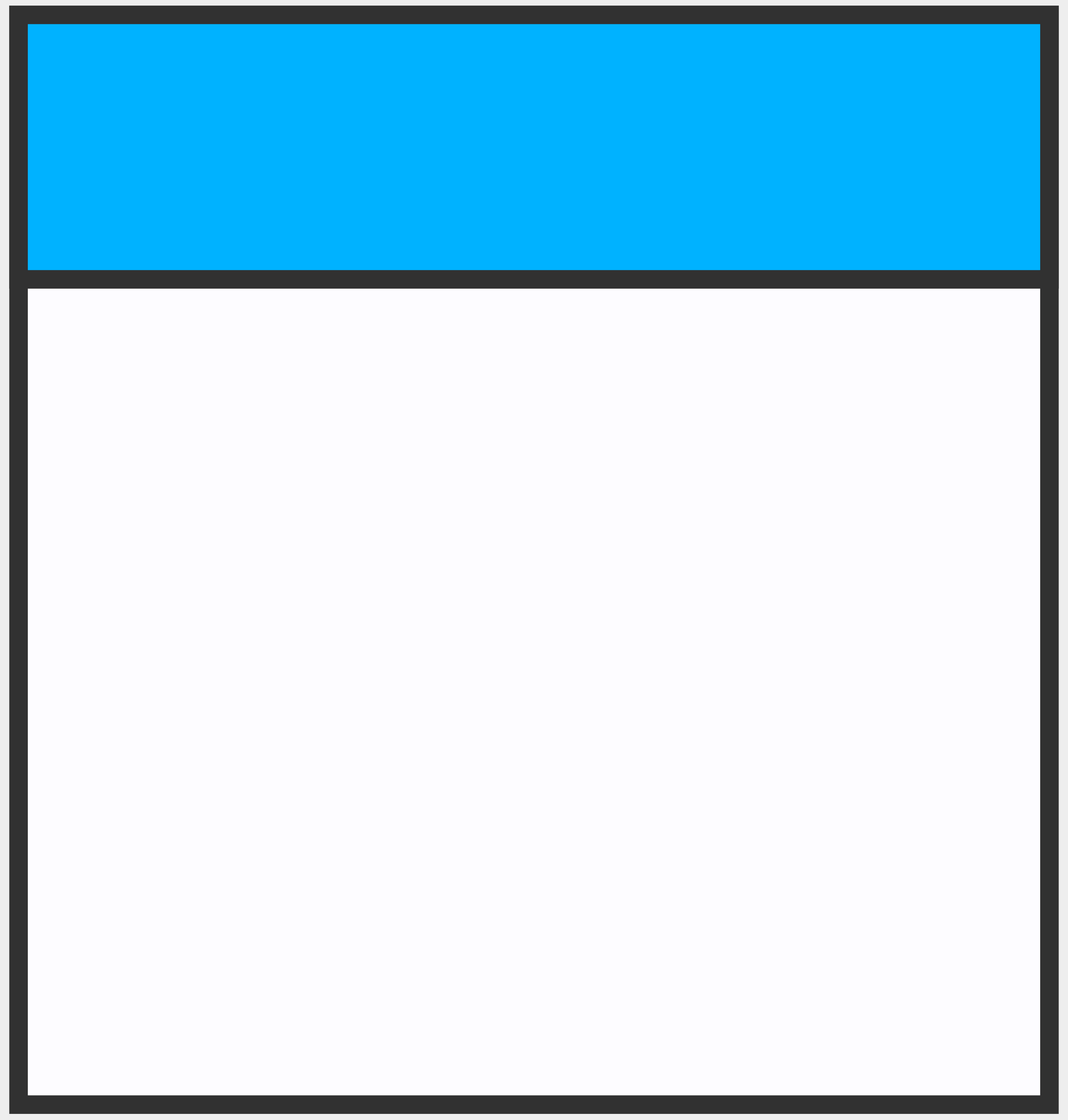
Initial goroutine stack size: **2048 bytes**

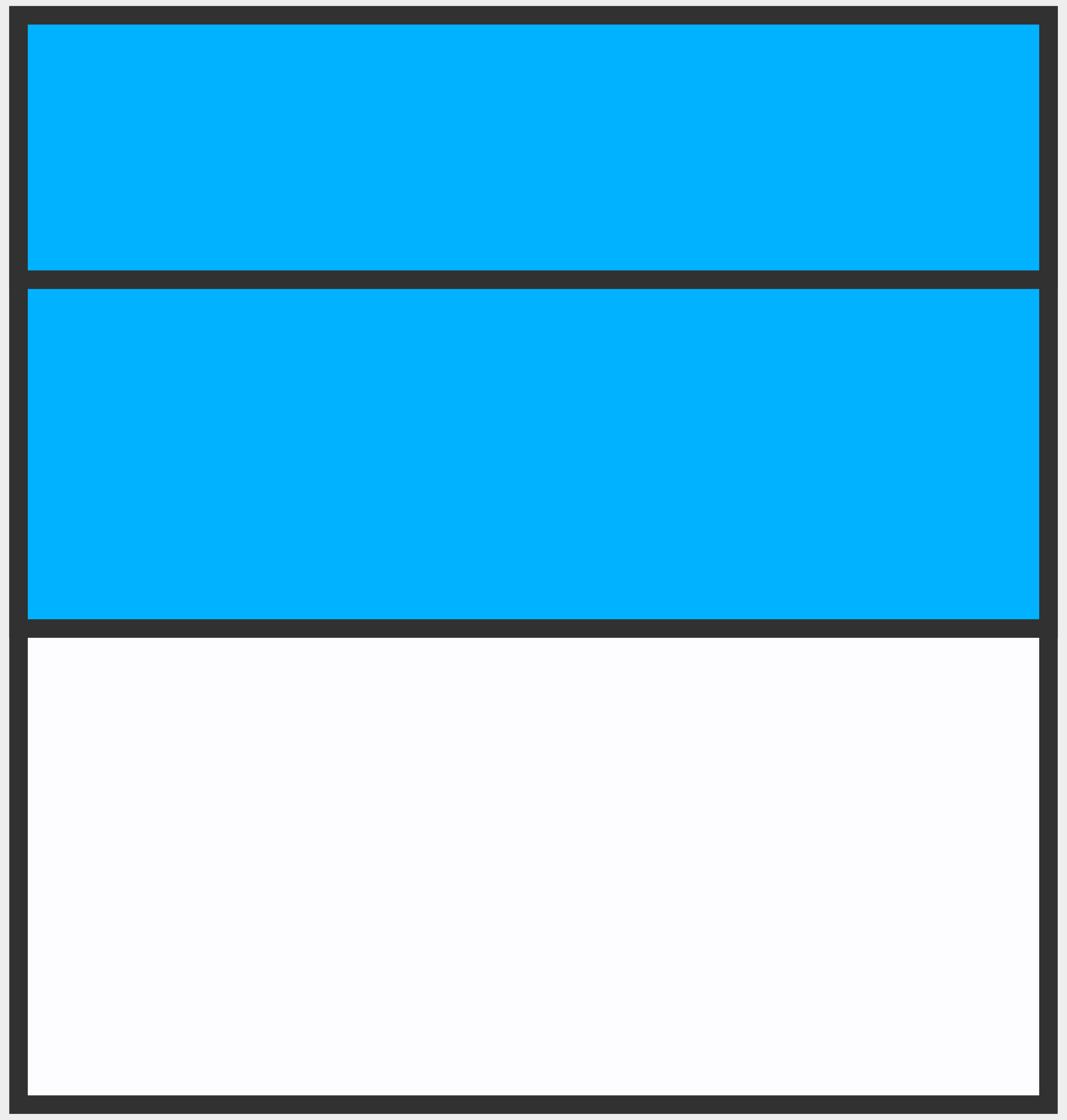
System stack size: **1–8 megabytes**

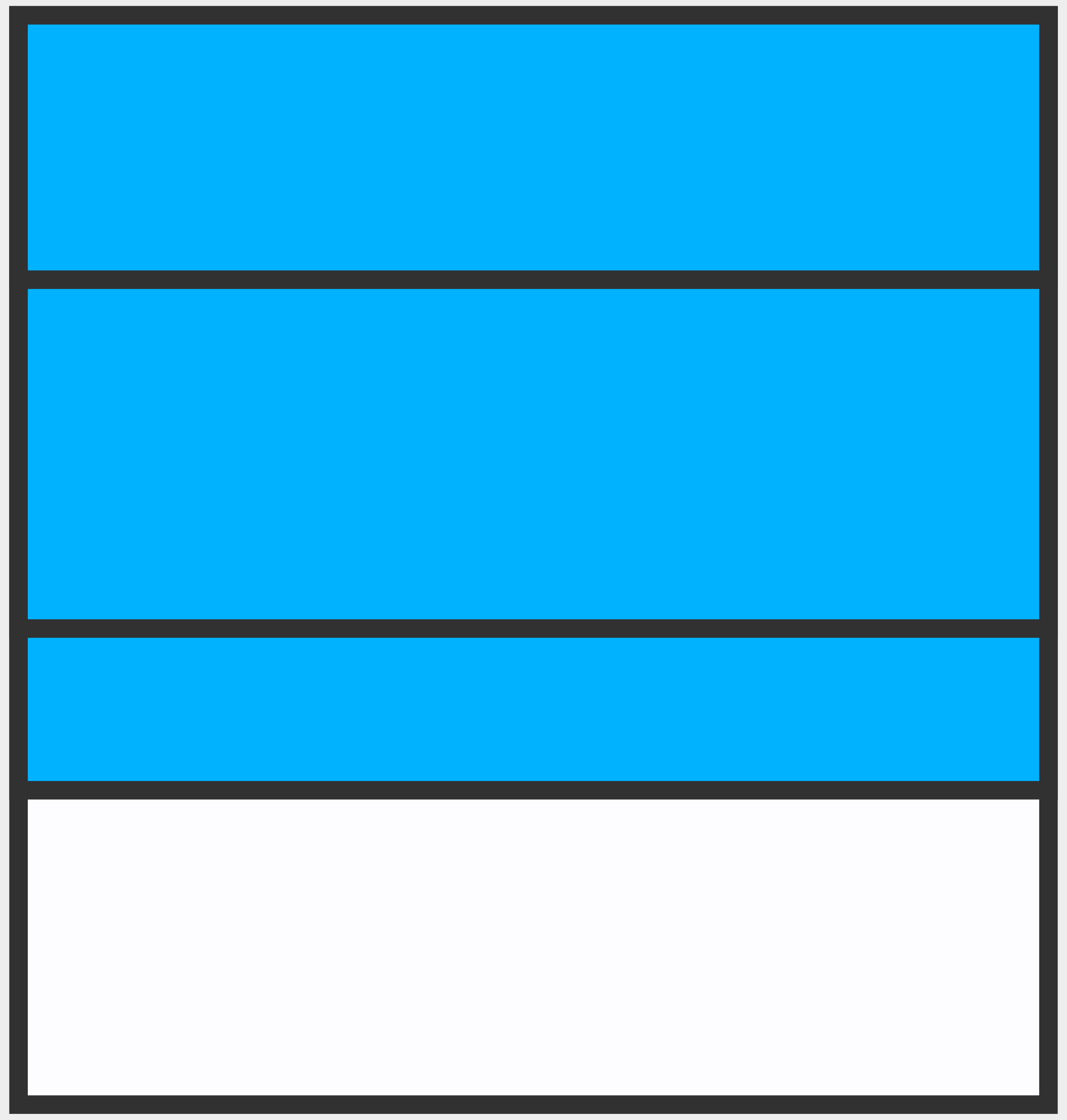


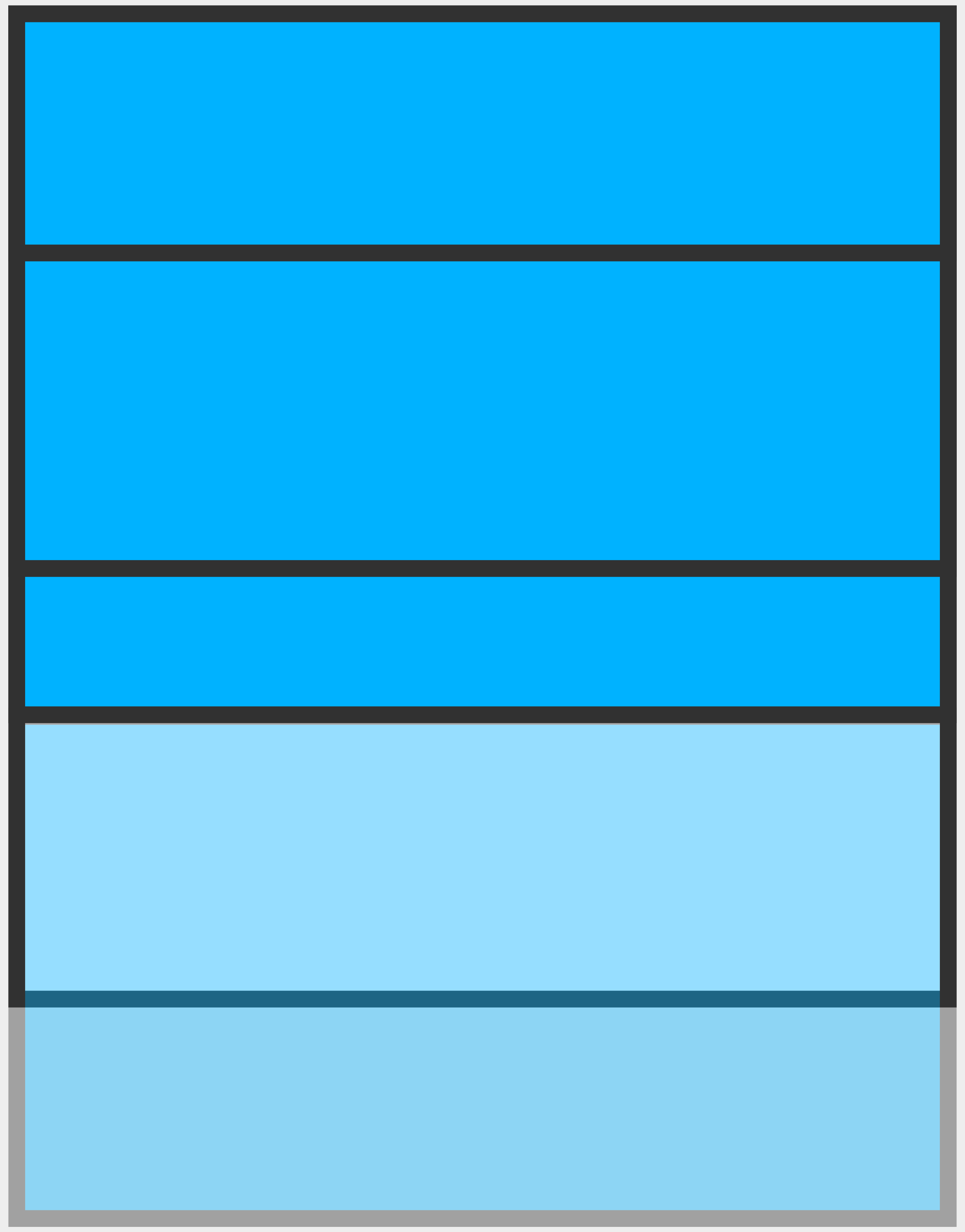
stack

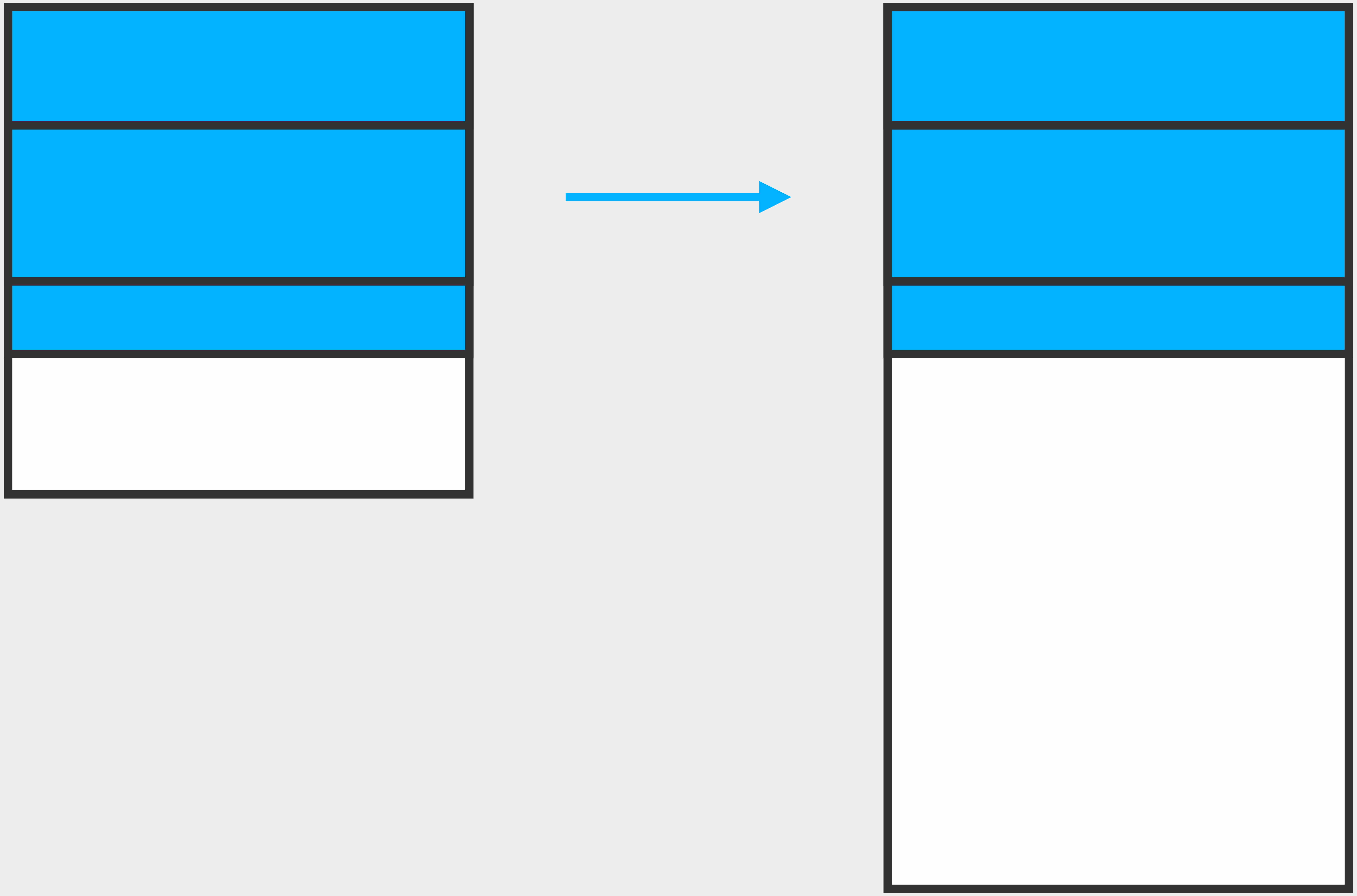
function frame

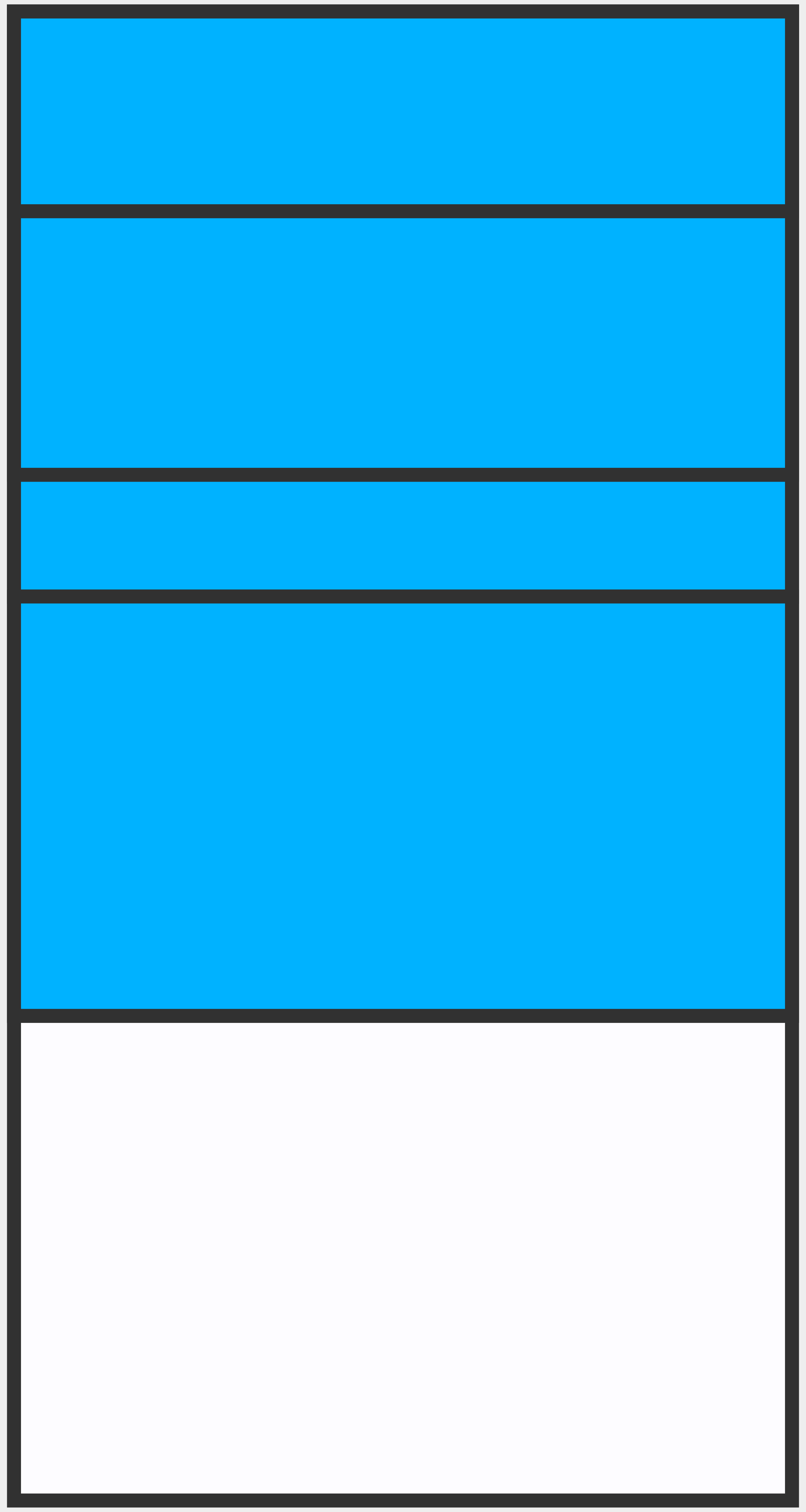












Function preamble

```
LEAQ framesize(SP), R0
CMPQ g→stackguard, R0
JHI 3(PC)
MOVQ m→morearg, $(argsize << 32)
CALL morestack(SB)
```

C doesn't call morestack

$\downarrow$

C code needs to run on a system stack

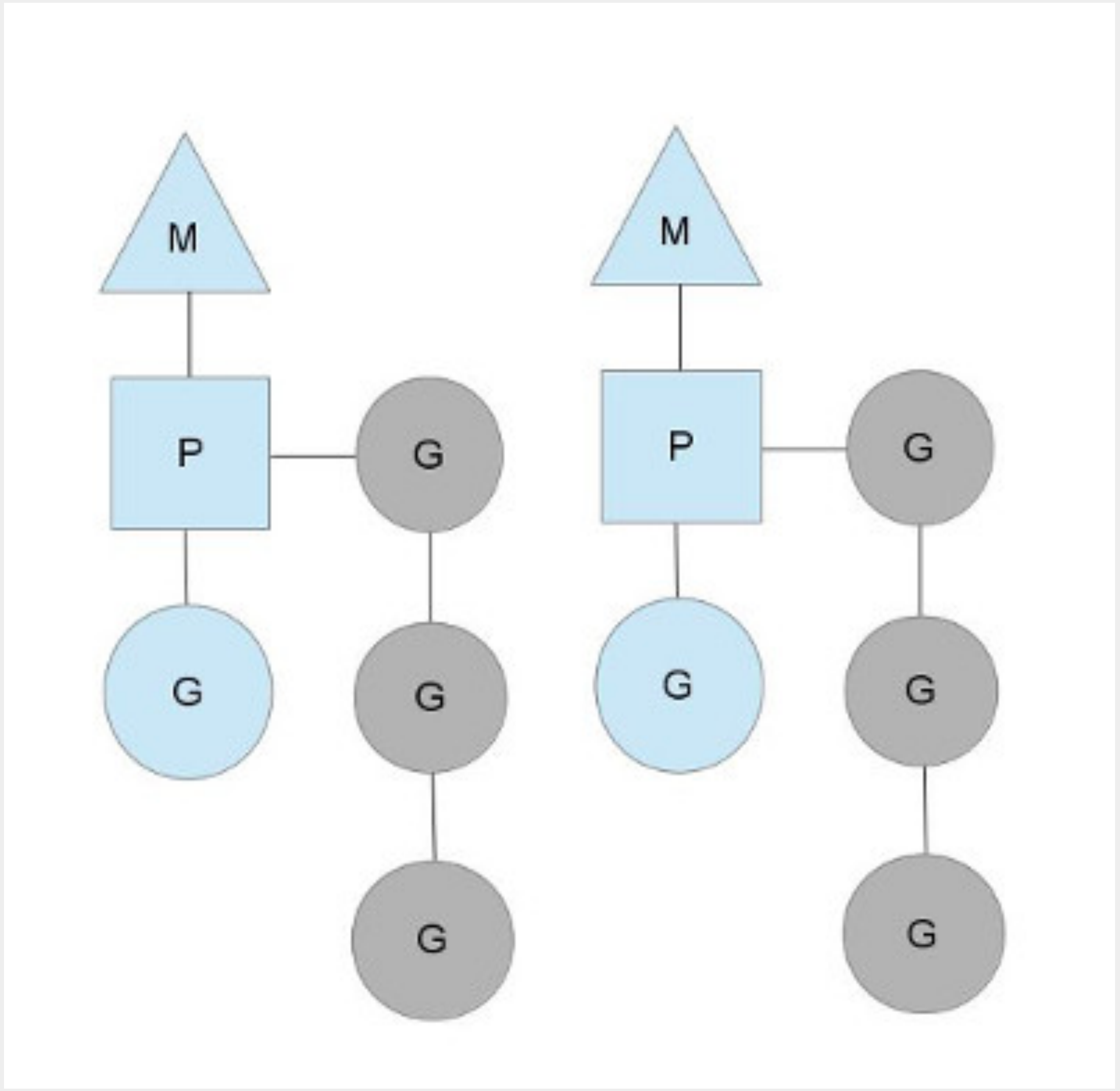
```
cgocall / asmcgocall
```

Learn more:

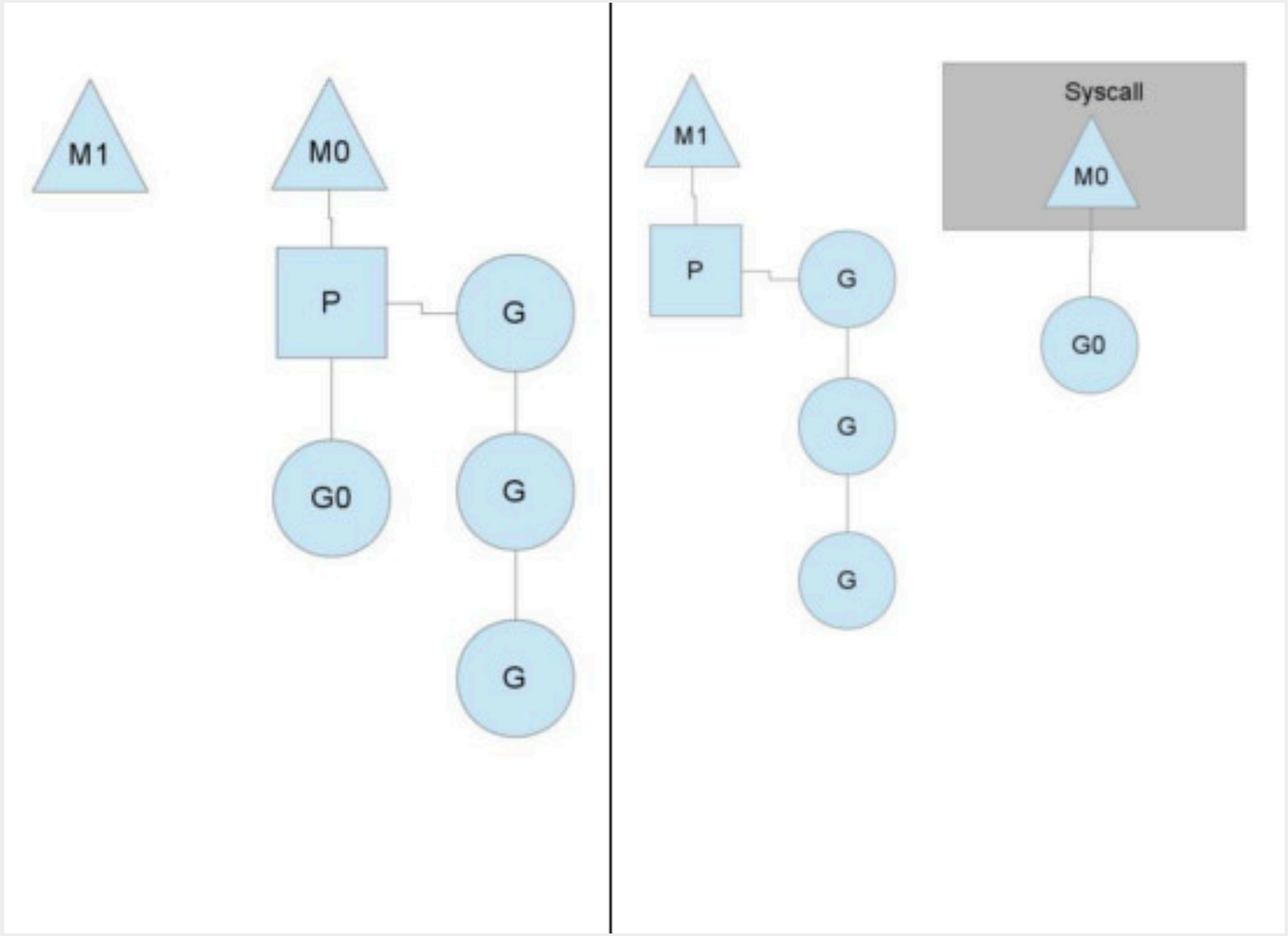
- src/runtime/stack.go
- src/runtime/cgocall.go
- How stacks are handled in Go by Daniel Morsing

  https://blog.cloudflare.com/how-stacks-are-handled-in-go/

Reason 3:
the scheduler



From https://morsmachine.dk/go-scheduler



From https://morsmachine.dk/go-scheduler

The Go scheduler is collaborative.

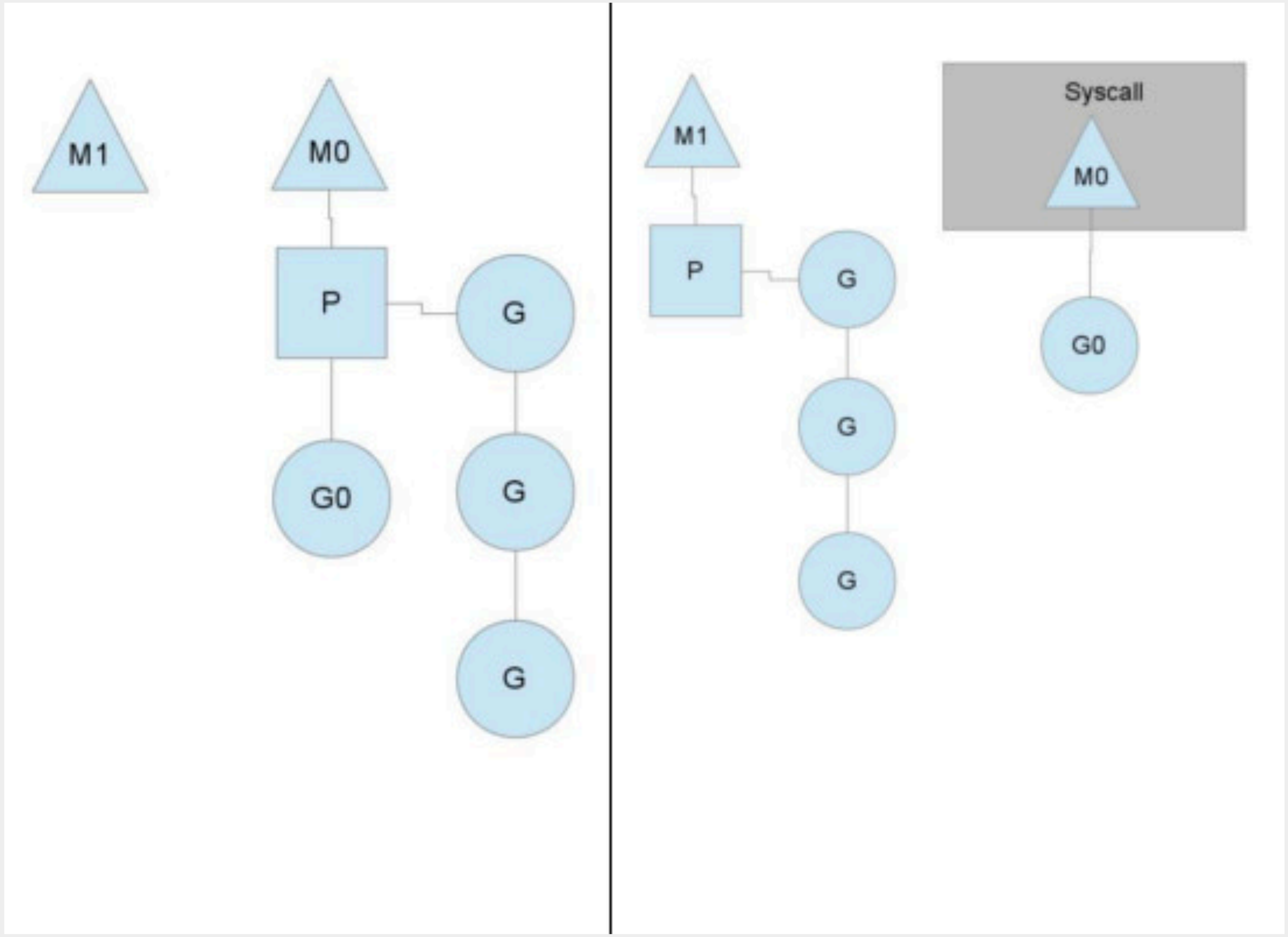
It can't preempt running code.

(ProTip: for {} is never what you want. Use select {}.)

```go
// Call from Go to C.
func cgocall(fn, arg unsafe.Pointer) int32 {
    // Announce we are entering a system call
    // so that the scheduler knows to create another
    // M to run goroutines while we are in the
    // foreign code.
    entersyscall()

    errno := asmcgocall(fn, arg)

    exitsyscall()
```

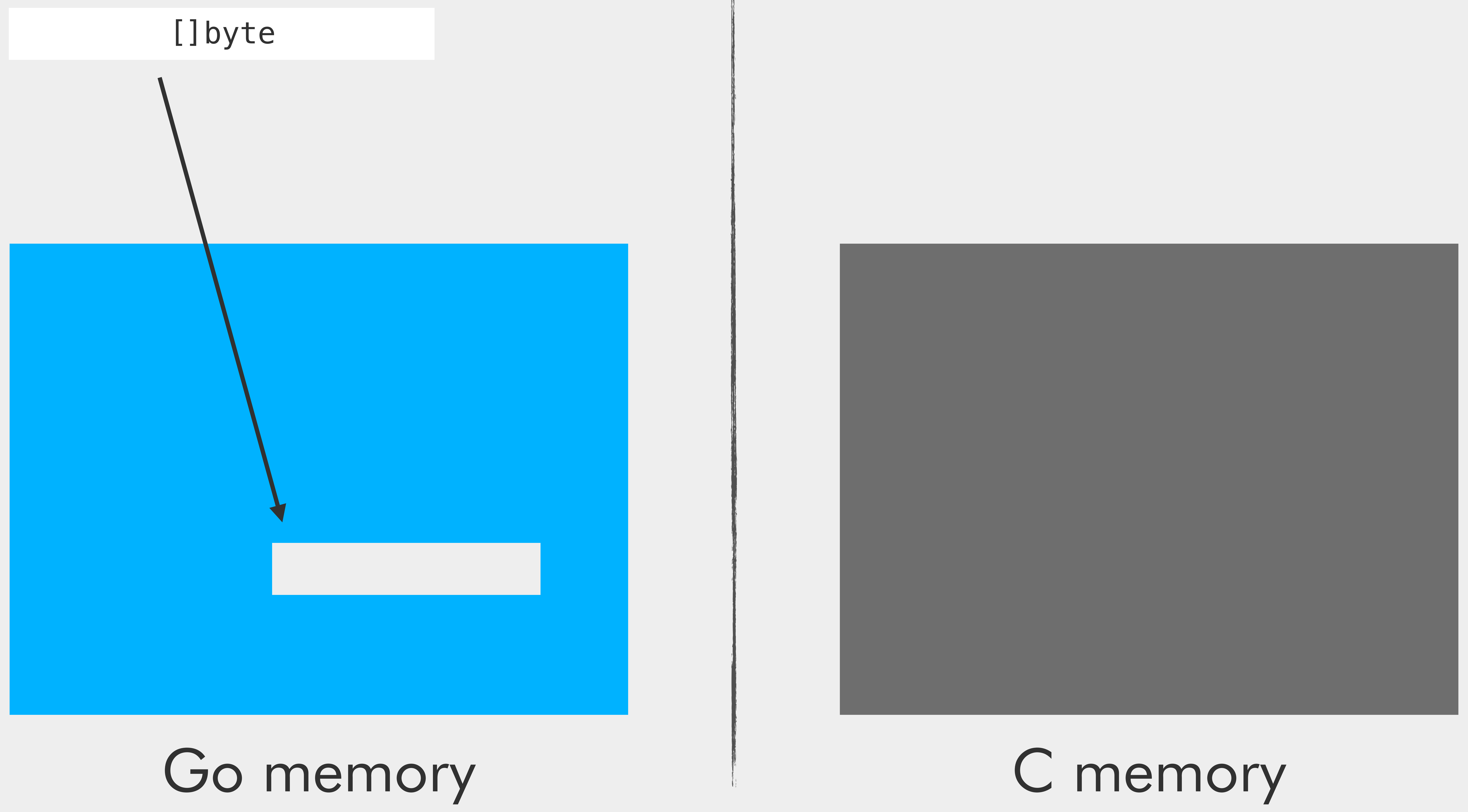


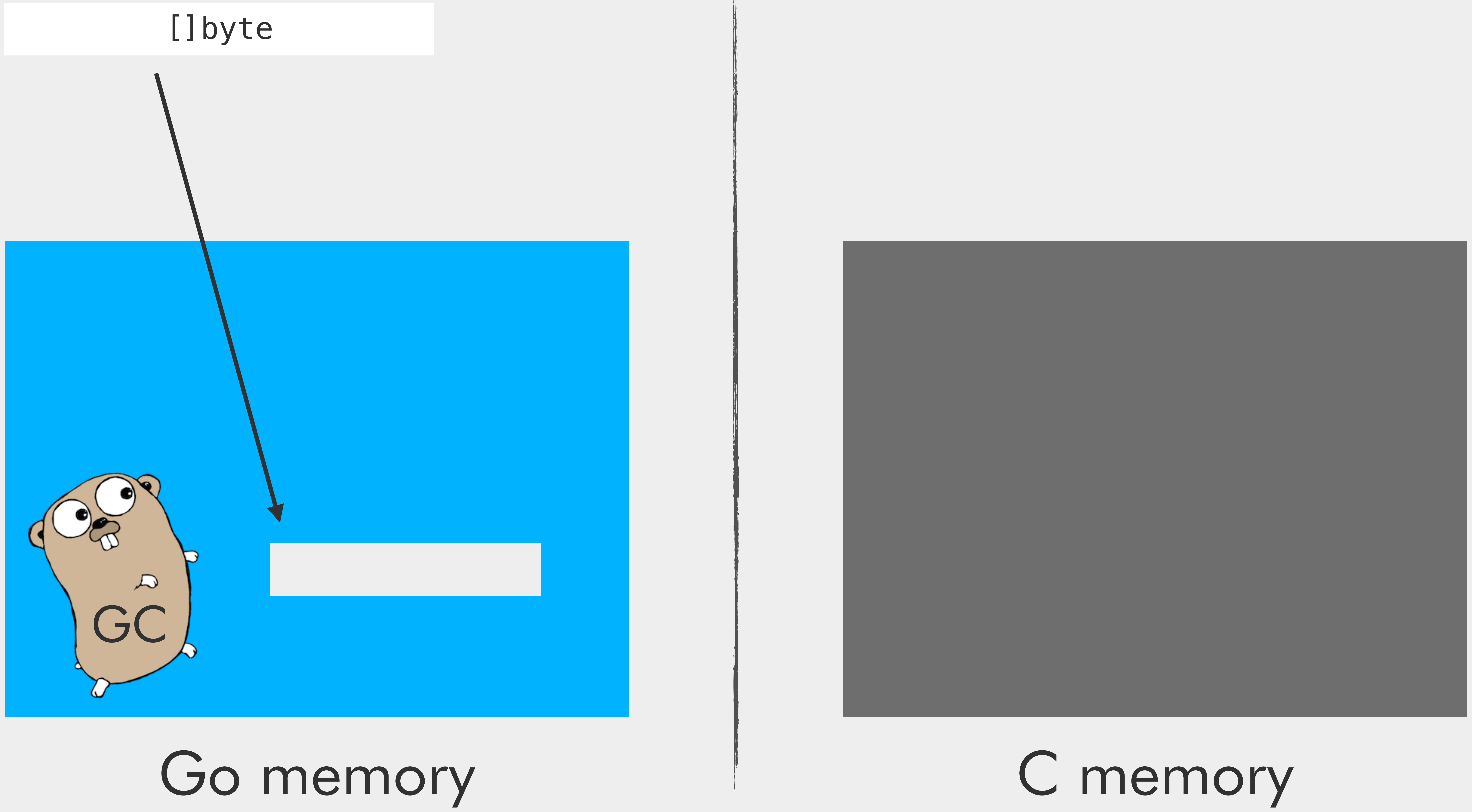
From https://morsmachine.dk/go-scheduler

Learn more:

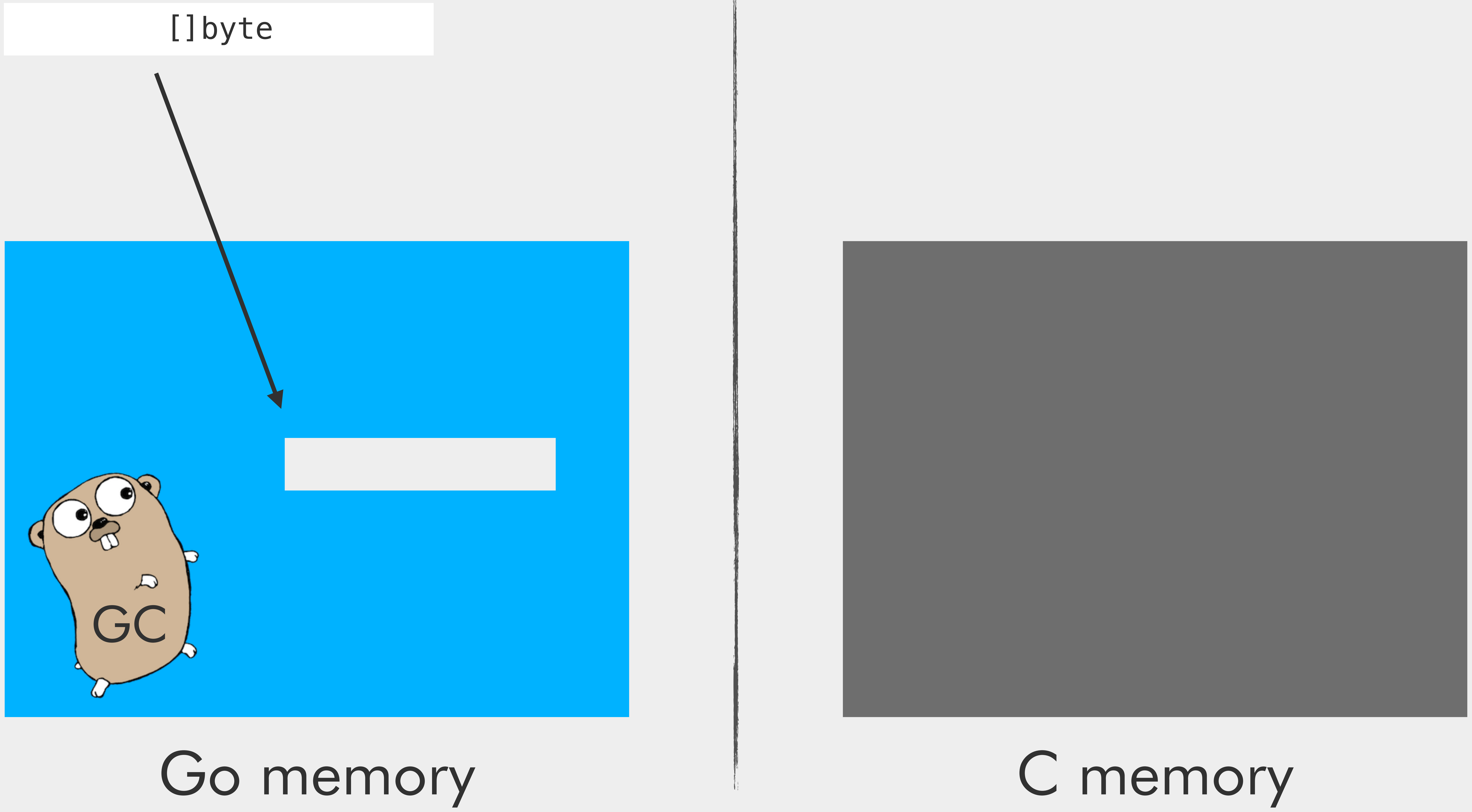
- src/runtime/proc.go → reentersyscall
- The Go scheduler by Daniel Morsing
  https://morsmachine.dk/go-scheduler
- Performance without the event loop by Dave Cheney
  https://dave.cheney.net/2015/08/08/performance-without-the-event-loop

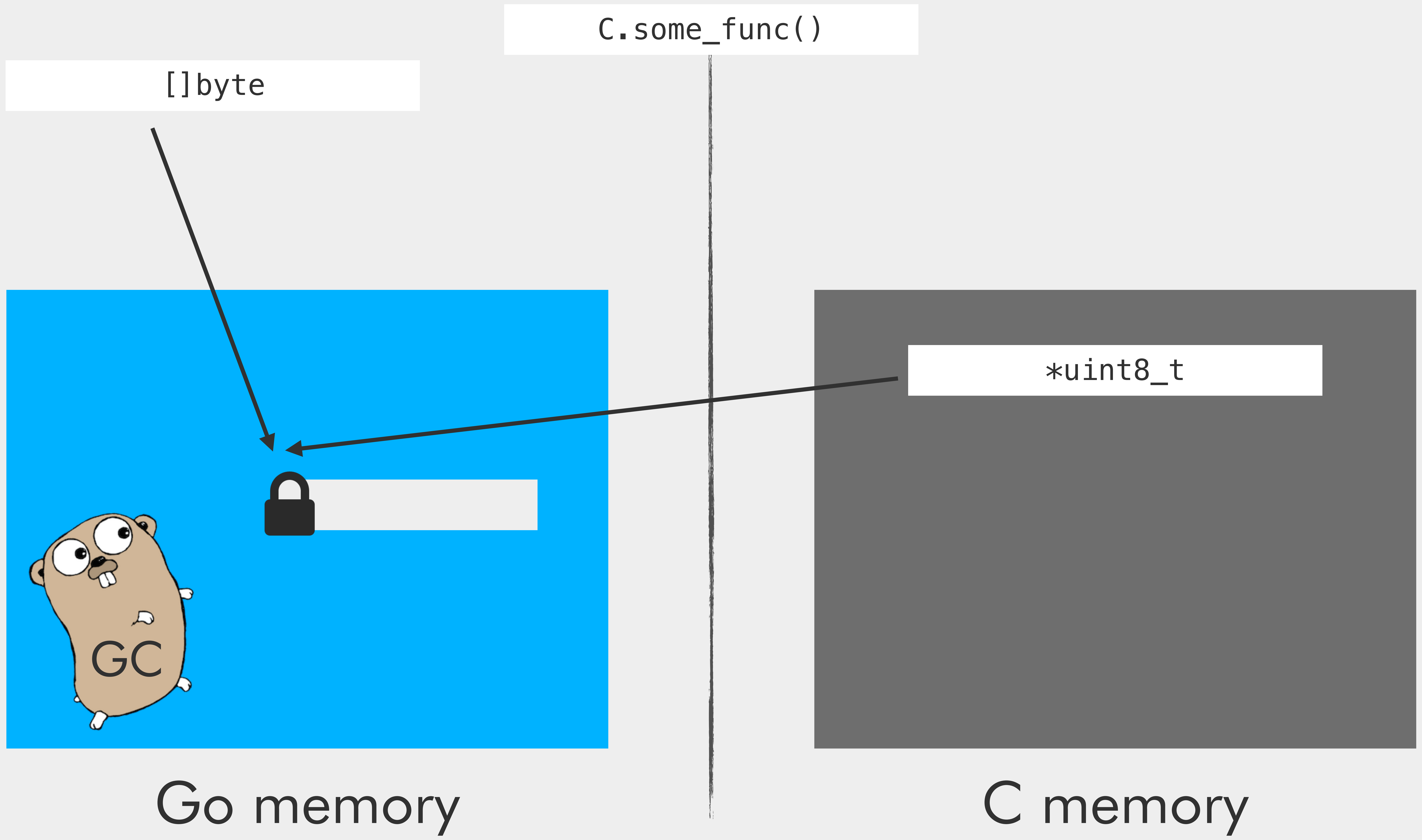
Reason 4:
the garbage collector


[]byte
Go memory
C memory

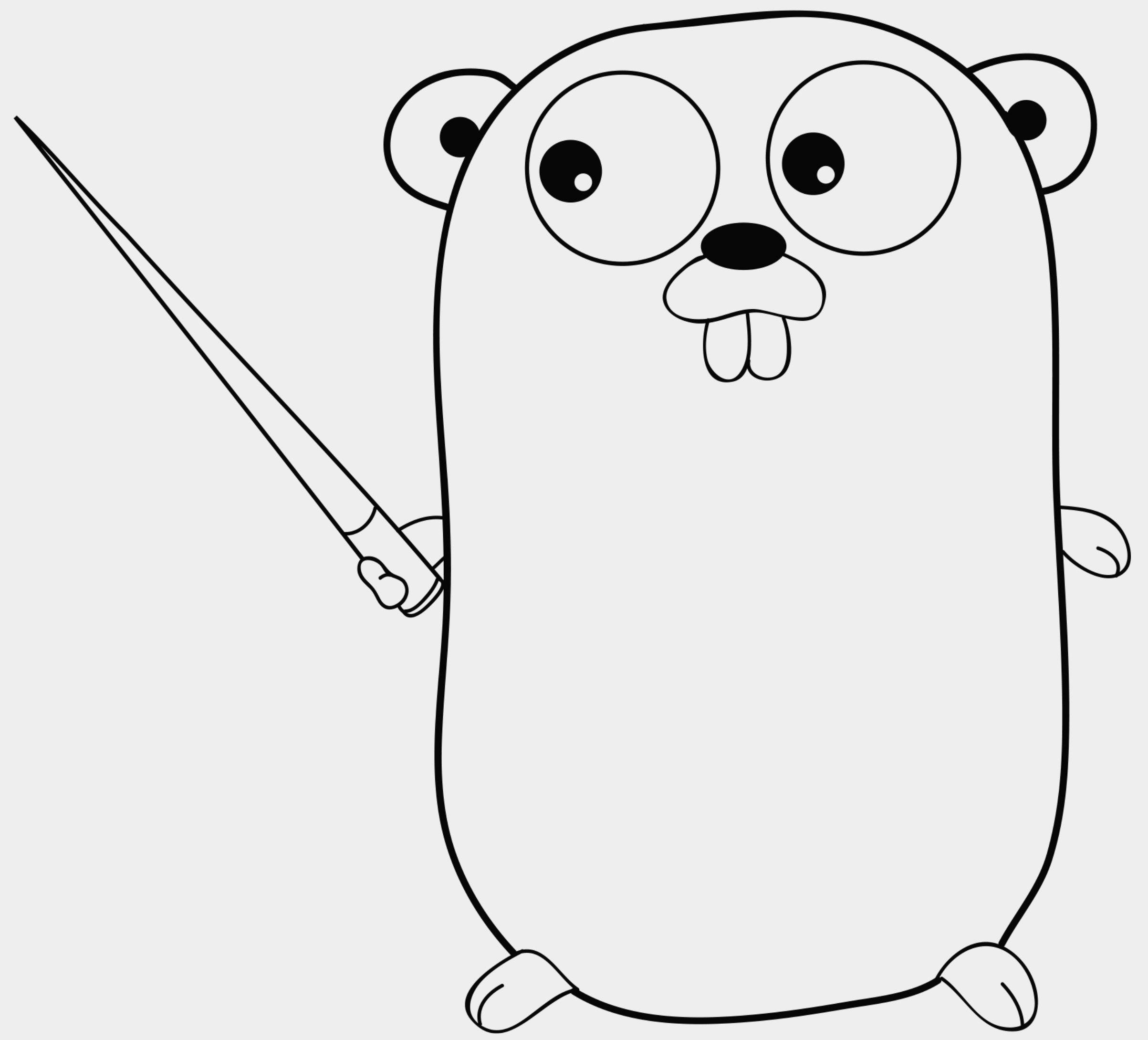


[]byte
GC
Go memory
C memory


[]byte
GC
Go memory
C memory


C.some_func()
[]byte
*uint8_t
GC
Go memory
C memory

The cgo rules



You may pass a Go pointer

… if it doesn’t point to other pointers

… and C can’t keep a reference to it

The GC must see all the Go pointers.

```
panic: runtime error: cgo argument
has Go pointer to Go pointer
```

```
GODEBUG=cgocheck=2
```

Learn more:

- From cgo back to Go @ GopherCon 2016

  https://speakerdeck.com/filosottile/from-cgo-back-to-go-gophercon-2016

Thank you!

filippo@golang.org

@FiloSottile