Introduction to Go Scheduler

Introduction o Go

- Very easy to use lots of light weight processes (Go routines) in the same time.
- Use the “go” keyword.

```
go func {
        time.Sleep(time.Second)
        fmt.Println("hello 1")
}
fmt.Println("hell 2")
```

Why Not Use System Scheduler

- Processes in an application don’t need too many context.
- It’s difficult for OS to handle too many threads or processes.
- System scheduler is too overhead.

What Will Include

- Basic structures
- The init of the scheduler
- The init of a Go routine
- The schedule that happens in the system call
- How to change current running Go routine

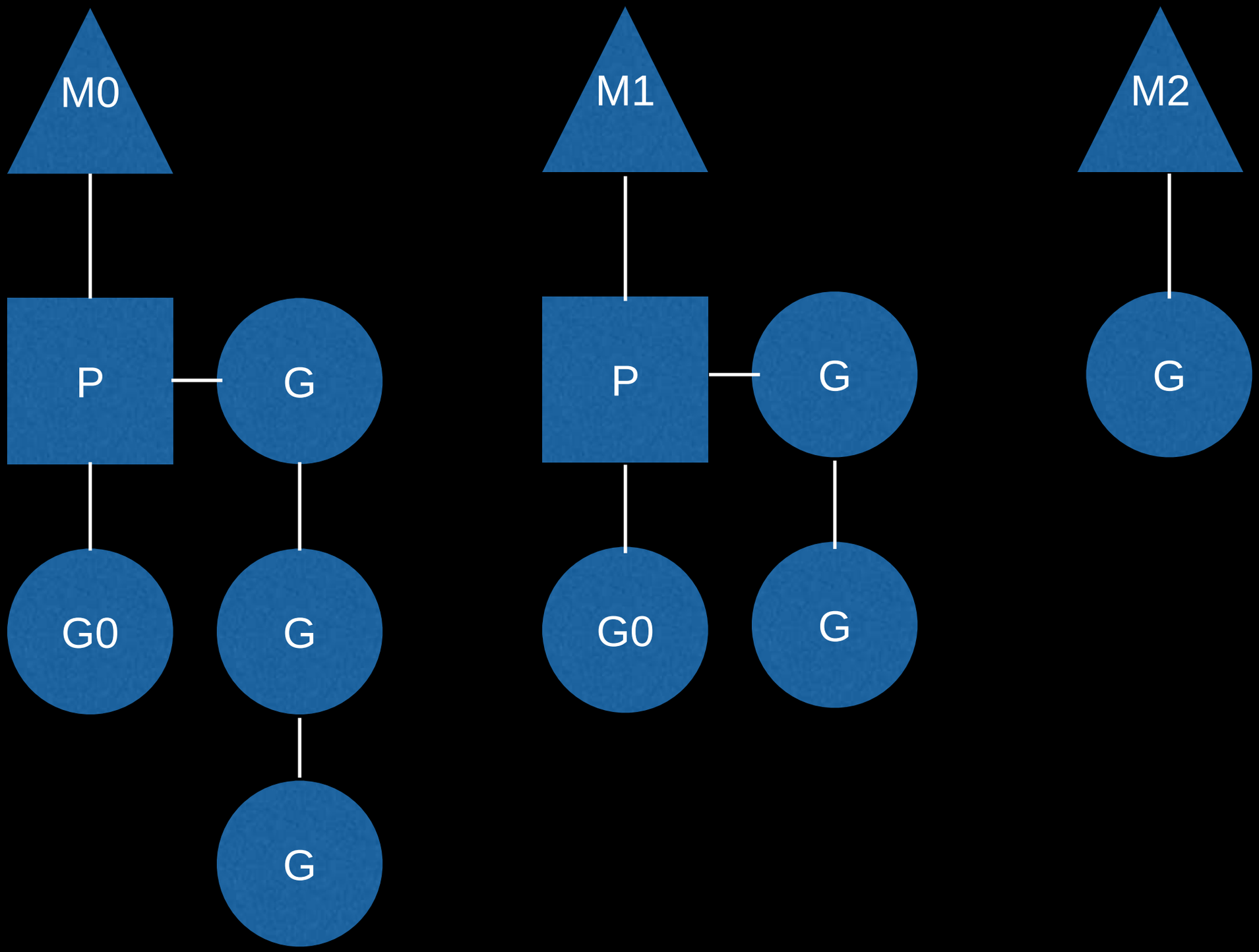
Source Code

- src/pkg/runtime/proc.c
- src/pkg/runtime/runtime.h
- src/pkg/runtime/asm_386.s

Basic Structures

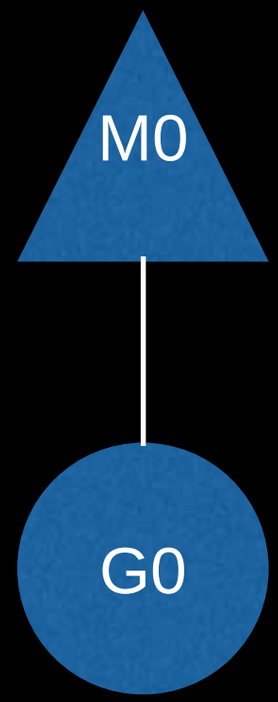
- M: OS threads.
- P: Context to run Go routines.
- G: Go routine.

Basic Structures

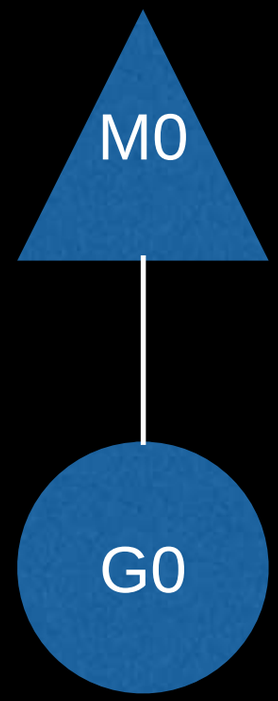


The Init of Scheduler

```
// set up m and g "registers"
get_tls(BX)
LEAL    runtime·g0(SB), CX
MOVL    CX, g(BX)
LEAL    runtime·m0(SB), AX

// save m->g0 = g0
MOVL    CX, m_g0(AX)
// save g0->m = m0
MOVL    AX, g_m(CX)

CALL    runtime·emptyfunc(SB)   // fault if

// convention is D is always cleared
CLD

CALL    runtime·check(SB)

// saved argc, argv
MOVL    120(SP), AX
MOVL    AX, 0(SP)
```

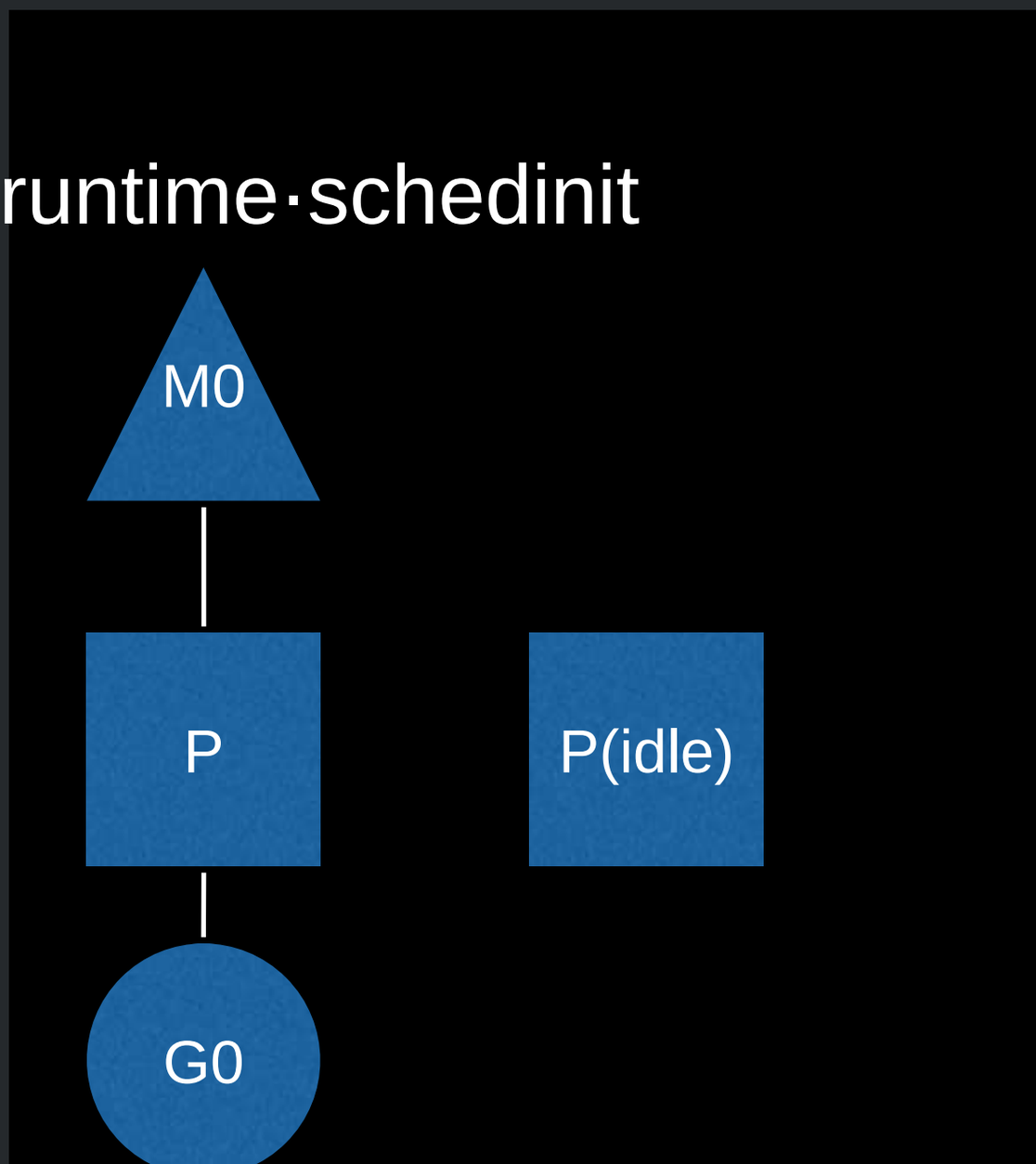


```
CLD

CALL    runtime·check(SB)

// saved argc, argv
MOVL    120(SP), AX
MOVL    AX, 0(SP)
MOVL    124(SP), AX
MOVL    AX, 4(SP)
CALL    runtime·args(SB)
CALL    runtime·osinit(SB)
CALL    runtime·hashinit(SB)
CALL    runtime·schedinit(SB)

// create a new goroutine to start program
PUSHL   $runtime·main·f(SB)     // entry
PUSHL   $0      // arg size
ARGSIZE(8)
CALL    runtime·newproc(SB)
ARGSIZE(-1)
POPL    AX
POPL    AX

// start this M
CALL    runtime·mstart(SB)
```



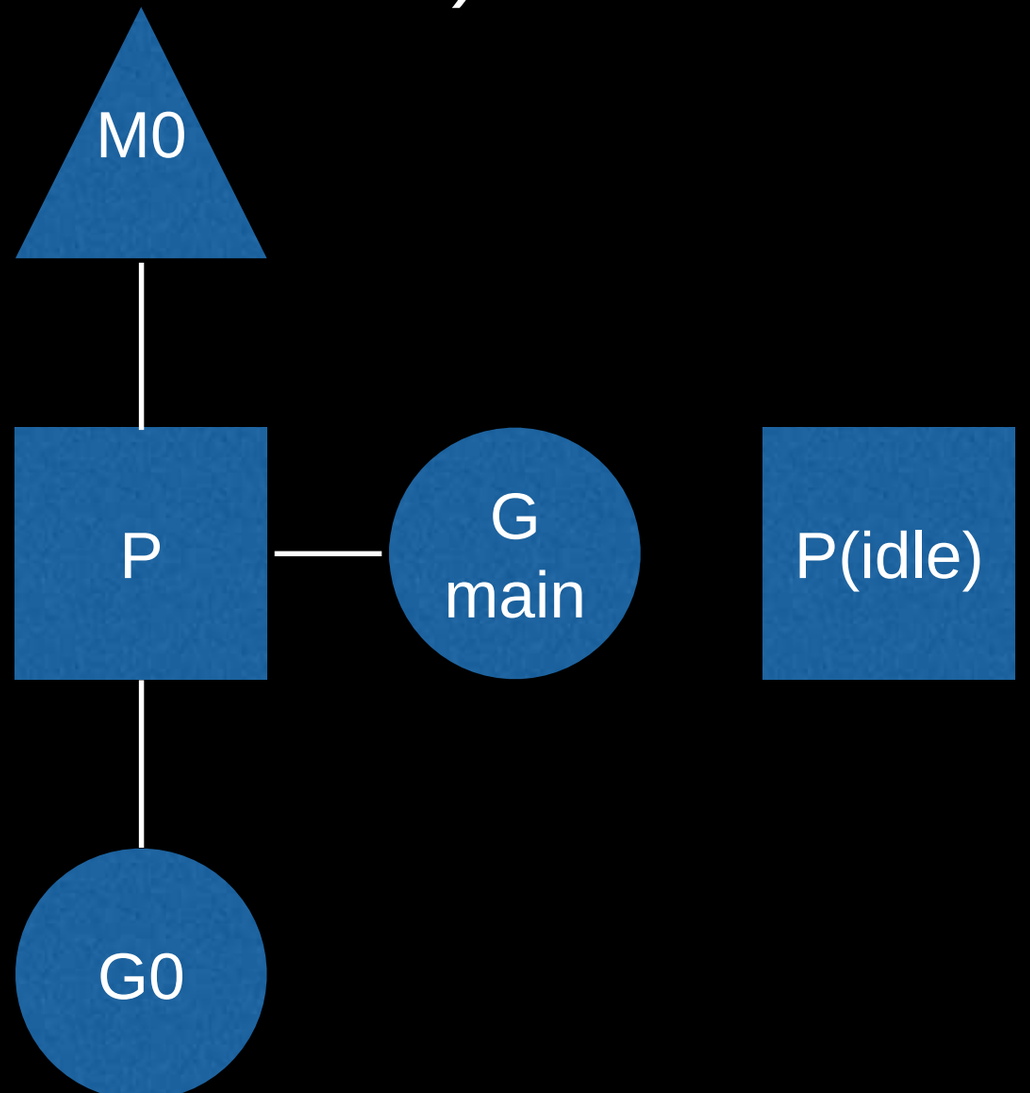
```
p = runtime·getenv("GOMAXPROCS");
if(p != nil && (n = runtime·atoi(p)) > 0) {
        if(n > MaxGomaxprocs)
                n = MaxGomaxprocs;
        procs = n;
}
runtime·allp = runtime·malloc((MaxGomaxprocs+1)*sizeof(runtime·allp[0]));
procresize(procs);
```

```
CALL    runtime·args(SB)
CALL    runtime·osinit(SB)
CALL    runtime·hashinit(SB)
CALL    runtime·schedinit(SB)

// create a new goroutine to start program
PUSHL   $runtime·main·f(SB)     // entry
PUSHL   $0      // arg size
ARGSIZE(8)
CALL    runtime·newproc(SB)
ARGSIZE(-1)
POPL    AX
POPL    AX

// start this M
CALL    runtime·mstart(SB)
```



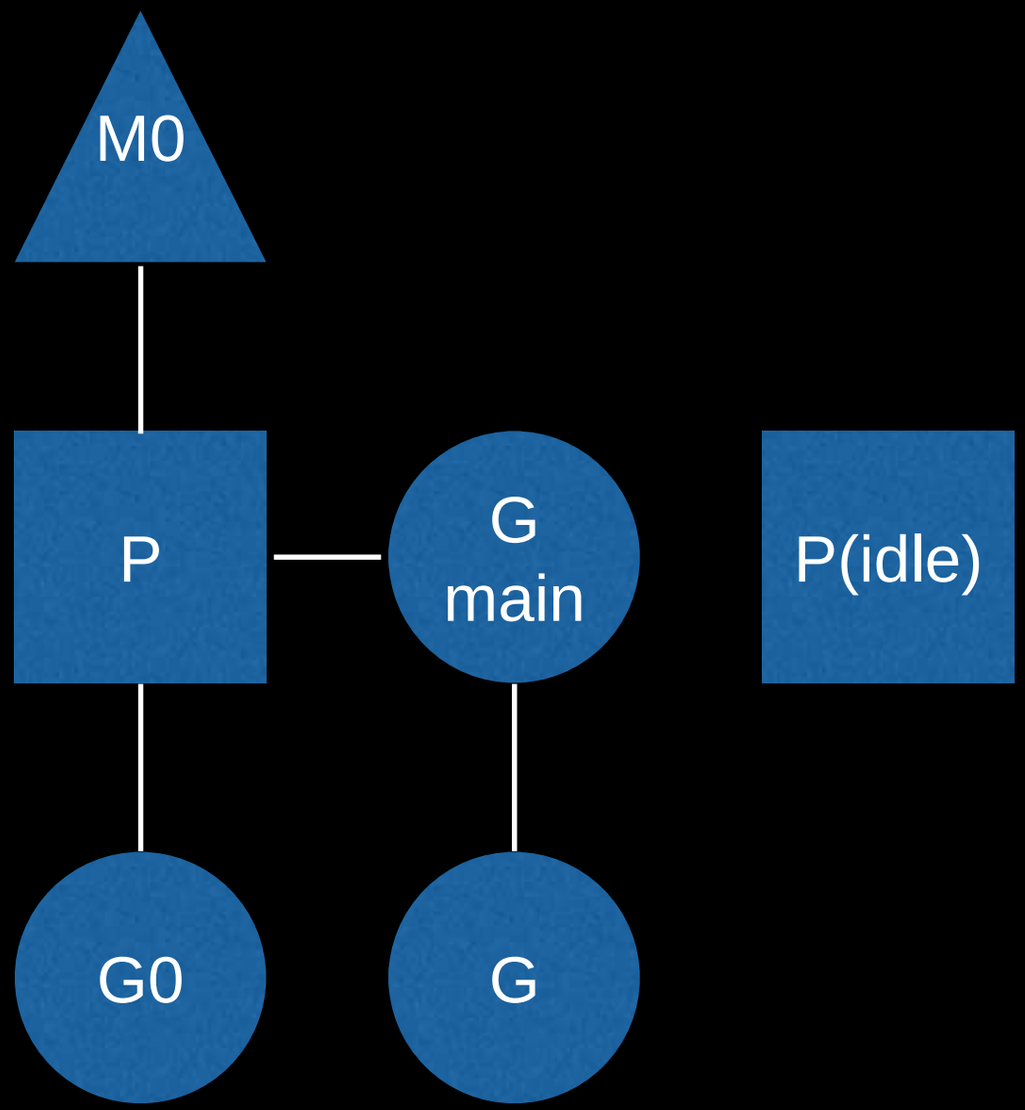
```
CALL    runtime·emptyfunc(SB)   // fault if

// convention is D is always cleared
CLD

CALL    runtime·check(SB)

// saved argc, argv
MOVL    120(SP), AX
MOVL    AX, 0(SP)
MOVL    124(SP), AX
MOVL    AX, 4(SP)
CALL    runtime·args(SB)
CALL    runtime·osinit(SB)
CALL    runtime·hashinit(SB)
CALL    runtime·schedinit(SB)

// create a new goroutine to start program
PUSHL   $runtime·main·f(SB)     // entry
PUSHL   $0      // arg size
ARGSIZE(8)
CALL    runtime·newproc(SB)
ARGSIZE(-1)
POPL    AX
POPL    AX

// start this M
CALL    runtime·mstart(SB)
```

runtime·main
main·main
runtime.newproc
runtime·mstart
(run the scheduler to execute G)



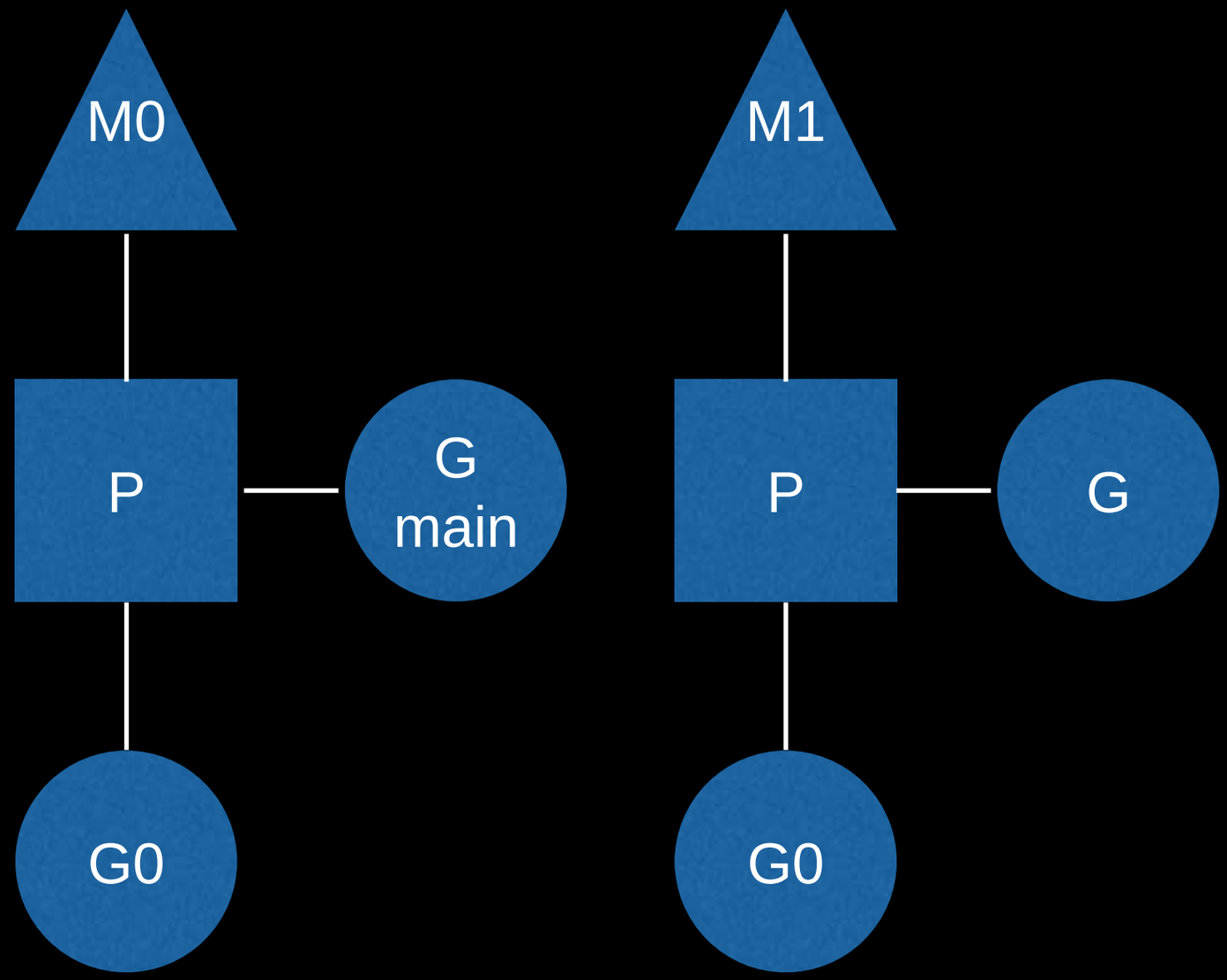
Init of Another Go Routine

- newproc
- newproc1



Init of Another Go Routine

- newproc
- newproc1
- wakep
- startm(nil, true)
- newm
- mstart



When to Schedule

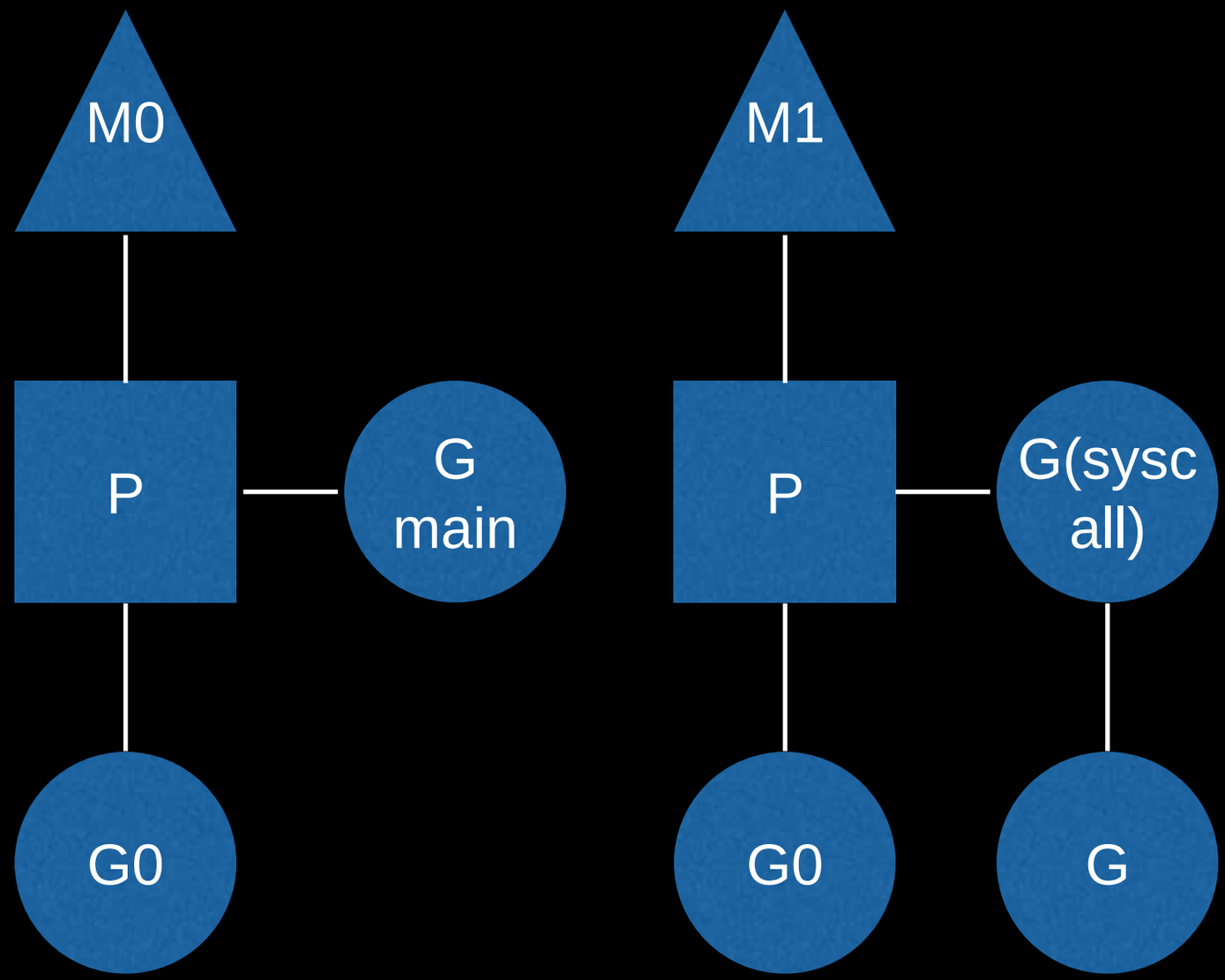
- block system call

After Go 1.2:

- call function
- use a channel

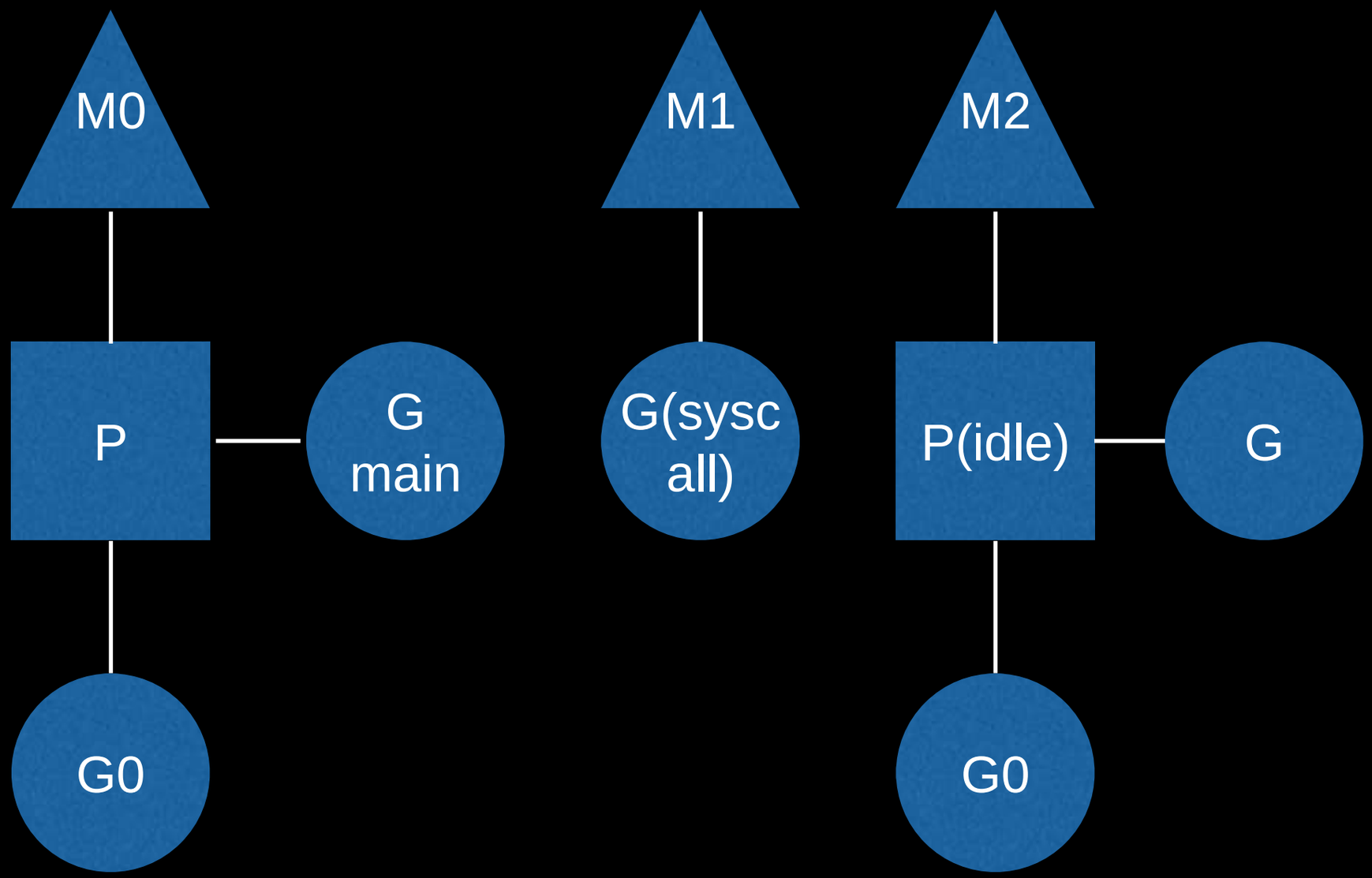
System Call

- .entersyscallblock



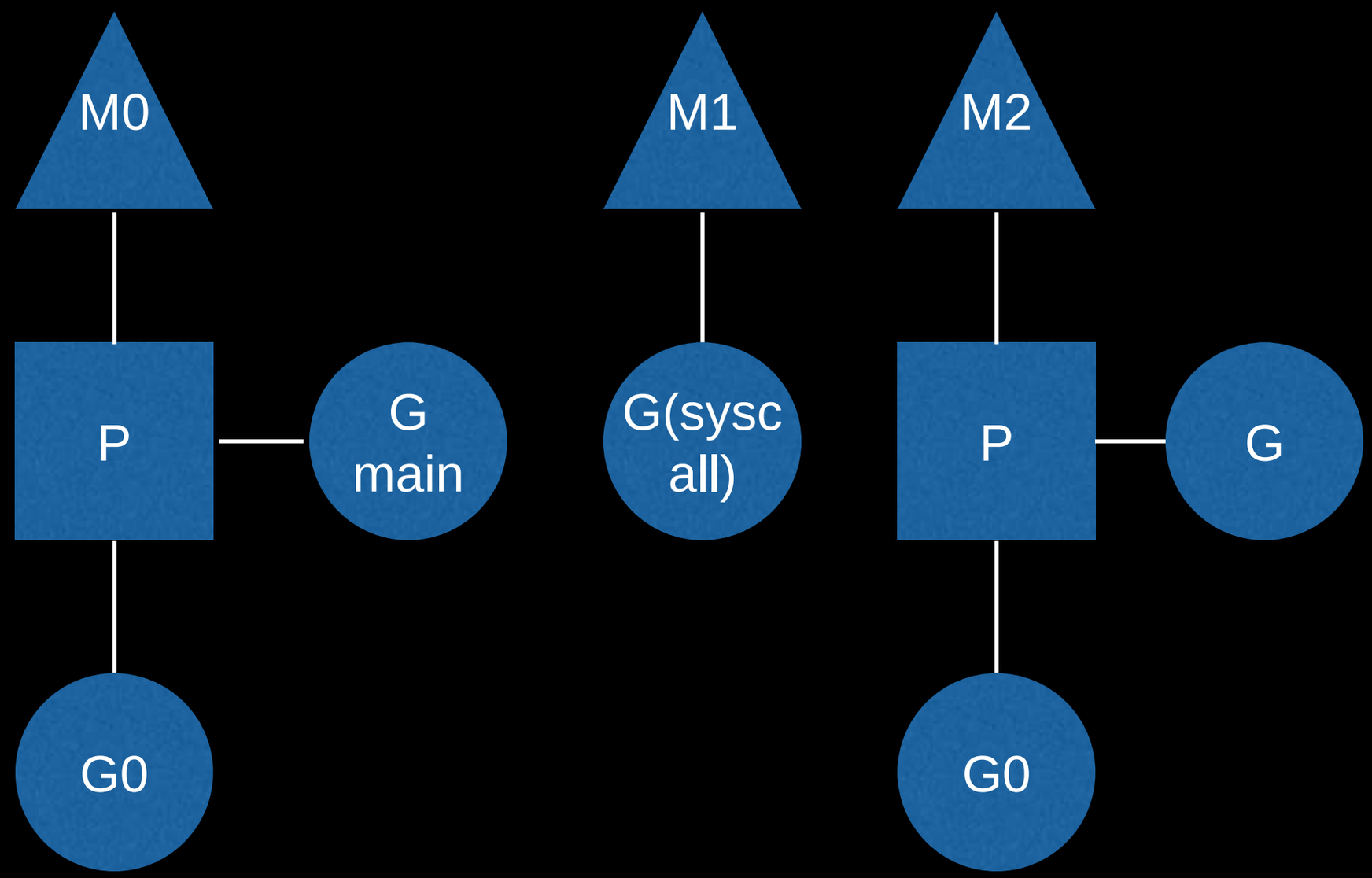
System Call

- .entersyscallblock
- handoffp



System Call

- .entersyscallblock
- handoffp
- startm(p



H

```
// void mcall(void (*fn)(G*))
// Switch to m->g0 stack, call fn(g).
// Fn must never return.  It should gogo(&g->sched)
// to keep running g.
TEXT runtime·mcall(SB), NOSPLIT, $0-4
        MOVL    fn+0(FP), DI

        get_tls(CX)
        MOVL    g(CX), AX       // save state in g->sched
        MOVL    0(SP), BX       // caller's PC
        MOVL    BX, (g_sched+gobuf_pc)(AX)
        LEAL    4(SP), BX       // caller's SP
        MOVL    BX, (g_sched+gobuf_sp)(AX)
        MOVL    AX, (g_sched+gobuf_g)(AX)

        // switch to m->g0 & its stack, call fn
        MOVL    g(CX), BX
        MOVL    g_m(BX), BX
        MOVL    m_g0(BX), SI
        CMPL    SI, AX  // if g == m->g0 call badmcall
        JNE     3(PC)
        MOVL    $runtime·badmcall(SB), AX
        JMP     AX
        MOVL    SI, g(CX)       // g = m->g0
        MOVL    (g_sched+gobuf_sp)(SI), SP      // sp = m->g0->sched.sp
        PUSHL   AX
        CALL    DI
        POPL    AX
        MOVL    $runtime·badmcall2(SB), AX
        JMP     AX
        RET
```

```go
type Counter struct {
	Count int
	Mu    sync.Mutex
}

func loop1() {
	loop1()
}

func loop2() {
	for {
	}
}

func main() {
	counter := Counter{Count: 0}
	loop2()

	for j := 0; j < 10; j++ {
		go func() {
			// get uid
			counter.Mu.Lock()
			id := counter.Count
			counter.Count += 1
			counter.Mu.Unlock()

			fmt.Println(id)

			loop1()
		}()
	}
	fmt.Println("hi")
	time.Sleep(time.Second * 1000)
}
```

A Demo

A Demo

Run with flags:

- GODEBUG=schedtrace=1000,scheddetail=1
- GOMAXPROCS=4

Q & A