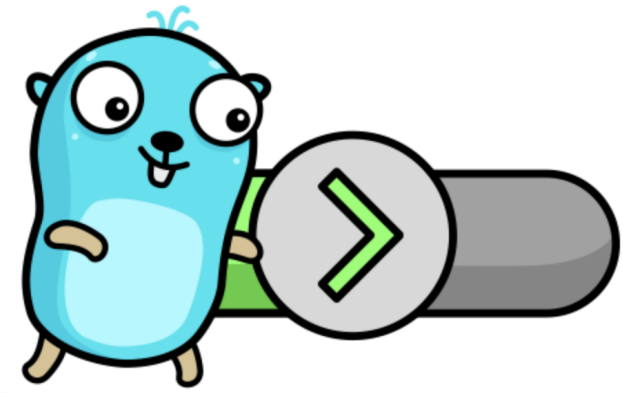
defer: оптимизации к Go 1.18

Этот небольшой урок посвящён не столько `defer`, сколько вопросу оптимизаций рантайма в Go в принципе.

Мы хотим лишний раз напомнить, что core-разработчики от релиза к релизу совершенствуют (или по крайней мере стараются) рантайм языка, стараясь увеличить производительность программ на Go.



Что изменилось?

К **Go 1.18** произошло много изменений:

- https://go.dev/doc/go1.15#compiler
- https://go.dev/doc/go1.16#compiler
- https://go.dev/doc/go1.17#compiler
- https://go.dev/doc/go1.18#compiler

Особенно крупными из которых (относящимися к нашей теме) можно считать:

- передачу параметров в функцию (и получение результатов из неё) через регистры, а не через стек как ранее – так называемая **register-based calling convention** (тыц);

- возможность встраивания большего количества функций.

Go 1.15 reduces typical binary sizes by around 5% compared to Go 1.14 by eliminating certain types of GC metadata and more aggressively eliminating unused type metadata.

The compiler can now inline functions with non-labeled `for` loops, method values, and type switches. The inliner can also detect more indirect calls where inlining is possible.

Go 1.17 implements a new way of passing function arguments and results using registers instead of the stack. Benchmarks for a representative set of Go packages and programs show performance improvements of about 5%, and a typical reduction in binary size of about 2%.

Functions containing closures can now be inlined.

The compiler now can inline functions that contain range loops or labeled for loops.

Вернёмся к вопросу производительности `defer` и запустим ещё раз бенчмарки:

```
$ go version
go version go1.18 darwin/arm64

$ go test -v -benchmem -bench .
Benchmark_withoutDefer         548712003        2.066 ns/op       0 B/op
0 allocs/op
Benchmark_withDefer            423015385        2.854 ns/op       0 B/op
0 allocs/op
Benchmark_withDefers           178489056        6.700 ns/op       0 B/op
0 allocs/op
Benchmark_withNotOpenDefer      34060629       34.27 ns/op       24 B/op
2 allocs/op
Benchmark_with8Defers           59411210       20.12 ns/op       0 B/op
0 allocs/op
```

```
Benchmark_with9Defers           18424676        65.42 ns/op       0 B/op
0 allocs/op
PASS
ok      examples/02-defer-statement/performance-2     8.518s
```

Кажется, что и обсуждать тут нечего? 🙂