Minimum Requirements

Operating Systems

Linux

Kernel version 2.6.23 or later. [*This depends on architecture though, we need to have specific builder for this.*] Linux/ARMv5 requires much newer kernels, at least v3.1 (for `__kuser_cmpxchg64`).

We don't support CentOS 5. The kernel is too old (2.6.18).

For little-endian MIPS64, kernel version 4.1 is known to fail, and 4.8 works.

If you are using tinyconfig (e.g. make tinyconfig) for embedded systems, you will also almost certainly enable printk in the kernel as well as a console; we will not include those generic options here. For Go, you must also enable CONFIG_FUTEX.

On arm64, an out of date (lower than version 2.33) ld.gold may cause shared library tests to fail (see https://github.com/golang/go/issues/28334).

Windows

For Go 1.10: Windows XP (w/ Service Pack 3) or higher.

For Go 1.11 and later: Windows Server 2008R2 and higher or Windows 7 and higher. We test on Windows Server 2008 R2, 2012 R2, and 2016, which are roughly Windows 7, Windows 8, and Windows 10.

macOS (née OS X, aka Darwin)

macOS Sierra 10.12 or higher requires Go 1.7.1 or above.

macOS Monterey 12 or higher requires Go 1.11 or above.

Go 1.15 and later only support macOS Sierra 10.12 or newer; see https://go.dev/doc/go1.15#darwin.

Go 1.17 and later only support macOS High Sierra 10.13 or newer; see https://go.dev/doc/go1.17#darwin.

We only have builders for 10.12, 10.14, 10.15, and 11.0 as of 2021-09-28.

OpenBSD

The current officially supported -stable versions only.

DragonFly BSD

Generally only the latest release version only. We have a builder, but it's not the most stable of our ports.

FreeBSD

FreeBSD 10 or higher. We only run builders testing FreeBSD 10.4, 11.2 and 12.0.

On arm64, at least 12.0 is required.

NetBSD

There are known NetBSD bugs (including kernel crashes) up to the current NetBSD 7.1. There is a reported fix in NetBSD 7.1.1 but it's unverified as of 2017-07-10, as we're not running builders again yet. See https://tip.golang.org/doc/go1.9#known_issues and https://github.com/golang/go/issues/20852

Native Client

Go 1.13: pepper_39 or newer.

Go 1.14 and later: unsupported.

Solaris

illumos (former OpenSolaris 10) based distributions or Oracle Solaris 11+.

iOS

iOS 12 or later.

Architectures

Microarchitecture support

For some architectures, Go supports compiling to specific microarchitectures using environment variables, e.g. GOAMD64 for go1.18 and later. Binaries will check at startup whether the requested microarchitecture level is supported. For example, a binary built with `GOAMD64=v3` will fail on a CPU that doesn't have LZCNT support.

Various microarchitecture levels and the environment variables used to select them are described with each architecture.

The build cache understands the microarchitecture environment variables and does not require any cleaning if you change them.

While performance is expected to improve when a higher minimum microarchitecture is requested, this might not be true in all cases. Benchmark your performance-critical code to verify performance improvements.

amd64

Until Go 1.17, the Go compiler always generated x86 binaries that could be executed by any 64-bit x86 processor.

Go 1.18 introduced 4 architectural levels for AMD64. Each level differs in the set of x86 instructions that the compiler can include in the generated binaries:

- GOAMD64=v1 (default): The baseline. Exclusively generates instructions that all 64-bit x86 processors can execute.
- GOAMD64=v2: all v1 instructions, plus CMPXCHG16B, LAHF, SAHF, POPCNT, SSE3, SSE4.1, SSE4.2, SSSE3.
- GOAMD64=v3: all v2 instructions, plus AVX, AVX2, BMI1, BMI2, F16C, FMA, LZCNT, MOVBE, OSXSAVE.
- GOAMD64=v4: all v3 instructions, plus AVX512F, AVX512BW, AVX512CD, AVX512DQ, AVX512VL.

Setting, for example, GOAMD64=v3, will allow the Go compiler to use AVX2 instructions in the generated binaries (which may improve performance in some cases); but these binaries will not run on older x86 processors that don't support AVX2.

The Go toolchain does not currently generate any AVX512 instructions.

Note that *processor* is a simplification in this context. In practice, support from the entire system (firmware, hypervisor, kernel) is needed.

See section Microarchitecture support for hints on how to use microarchitecture environment variables like GOAMD64.

386

See https://go.dev/doc/install/source#environment

- GO386=sse2 (default): Any processor with at least SSE2
- GO386=softfloat: All Pentium MMX or later processors (uses software floating point emulation)

arm

See https://go.dev/doc/install/source#environment

- GOARM=5: use software floating point; when CPU doesn't have VFP co-processor
- GOARM=6: use VFPv1 only; default if cross compiling; usually ARM11 or better cores (VFPv2 or better is also supported)
- GOARM=7: use VFPv3; usually Cortex-A cores

arm64

All ARMv8-A processors.

ppc64 (big endian)

POWER5 and above. Starting with Go 1.9, only POWER8 and above are supported.

ppc64le (little endian)

POWER8 and above.

mips64 (big endian)

MIPS III or higher. Builder is using MIPS64r2.

mips64le (little endian)

MIPS III or higher in little endian mode.

s390x

z196+

mips (big endian) and mipsle (little endian)

MIPS32r1

riscv64

rv64g (rv64imafd)

cgo

For programs using cgo, gcc 4.6 or newer is required.