# 测评 Go 编译器对 RISC-V 适配

PLCT Lab 每周技术分享

熊家辉
浙江工商大学

2024 年 8 月 23 日

# 目录

第 1 节

# Go

第 1 节

# Go

第 1.1 小节

## Go 简介

# Go 简介

Go（又称 Golang）是 Google 开发的一种静态强类型、编译型、并发型，并具有垃圾回收功能的编程语言。
罗伯特·格瑞史莫、罗勃·派克及肯·汤普逊于 2007 年 9 月开始设计 Go，稍后伊恩·兰斯·泰勒（Ian Lance Taylor）、拉斯·考克斯（Russ Cox）加入项目。Go 是基于 Inferno 操作系统所开发的。[5]Go 于 2009 年 11 月正式宣布推出，成为开放源代码项目，支持 Linux、macOS、Windows 等操作系统。
目前，Go 每半年发布一个二级版本（即从 a.x 升级到 a.y）。当前的版本是 1.23。

第 1 节

# Go

第 1.2 小节

## Go 的编译器

# Go 的编译器

- gc
- gccgo
- Gollvm

第 1 节

# Go

第 1.3 小节

## gc

# gc

Go 在 1.5 版本实现了自举，所使用的编译器在 `cmd/compile/` 中。这也是当前的默认实现。

# gc 编译流程：第一阶段：词法和语法分析

这部分代码位于 `cmd/compile/internal/syntax` 中。
在编译的第一阶段，源代码被 token 化（词法分析），解析（语法分析），并为每个源构造语法树文件。每个语法树都是相应源文件的精确表示对应于源的各种元素的节点，如表达式，声明和陈述。语法树还包括位置信息用于错误报告和调试信息的创建。
这一步会输出一个 AST 语法树。

# gc 编译流程：第二阶段：语义分析

这部分代码位于 `cmd/compile/internal/gc` 中。
对 AST 进行类型检查。第一步是名称解析和类型推断，它们确定哪个对象属于哪个标识符，以及每个表达式具有的类型。类型检查包括某些额外的检查，例如“声明和未使用”以及确定函数是否终止。
在 AST 上也进行了某些转换。一些节点基于类型信息被细化，例如从算术加法节点类型分割的字符串添加。另外还有死代码消除，函数调用内联和转义分析。

# gc 编译流程：第三阶段：SSA 生成

转换为 SSA 代码位于 `cmd/compile/internal/gc` 中。SSA 传递与规则代码位于 `cmd/compile/internal/ssa` 中。
在此阶段，AST 将转换为静态单一分配（SSA）形式，这是一种具有特定属性的低级中间表示，可以更轻松地实现优化并最终从中生成机器代码。

# gc 编译流程：第四阶段：机器码生成

底层 SSA 和架构特定的传递代码位于 `cmd/compile/internal/ssa` 中。生成机器码代码位于 `cmd/internal/obj` 中。
编译器的机器相关阶段以“底层”传递开始，该传递将通用值重写为其机器特定的变体。例如，在 amd64 存储器操作数上是可能的，因此可以组合许多加载存储操作。

# gc 优势

开箱即用，点击就送，无需动脑。

# gc 劣势

开箱即用，点击就送，无需动脑。

第 1 节

# Go

第 1.4 小节

## gccgo

# gccgo

gccgo 是 gcc 的 go 语言编译器。

# gccgo 优势

- 几乎所有的目标，包括 thead 支持
- 更小的文件
- 针对目标处理器的特定优化。

# gccgo 劣势

- 相对配置复杂（想想好多的编译选项）
- 没有默认附带，得自己搓。
- 支持 1.18 及以下的库，部分新库没有实现
- musl 和某些奇特的 libc 实现无法链接。

# 谢谢