基于链接器的RISC-V字加载指令优化

皇家墨尔本理工大学 乌鑫龙

6/26/2022

RISCV指令集简介

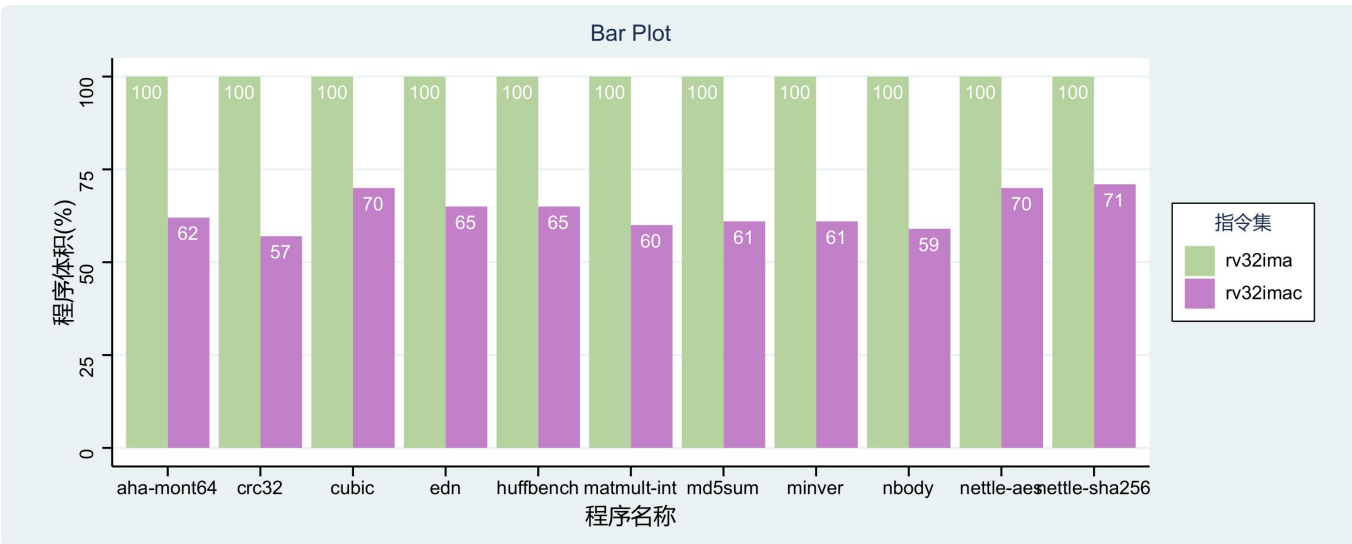
- **精简指令集：** 只定义常用指令
- **模块化：** 包含一个必选的基本整数指令集和多个可选的扩展指令集
  - 基本整数指令集（I扩展）：包含程序运行所需的基本指令，如整数/逻辑运算，内存访问，分支跳转以及CSR指令等
  - 整数乘法指令集（M扩展）
  - 原子指令集（A扩展）
  - 单精度浮点指令集（F扩展）
  - 双精度浮点指令集（D扩展）
  - 压缩指令集（C扩展）
  - ...

RISCV程序的二进制体积

- 二进制程序体积偏大

  复杂操作 -> 简单指令的组合

  压缩指令集（C扩展）

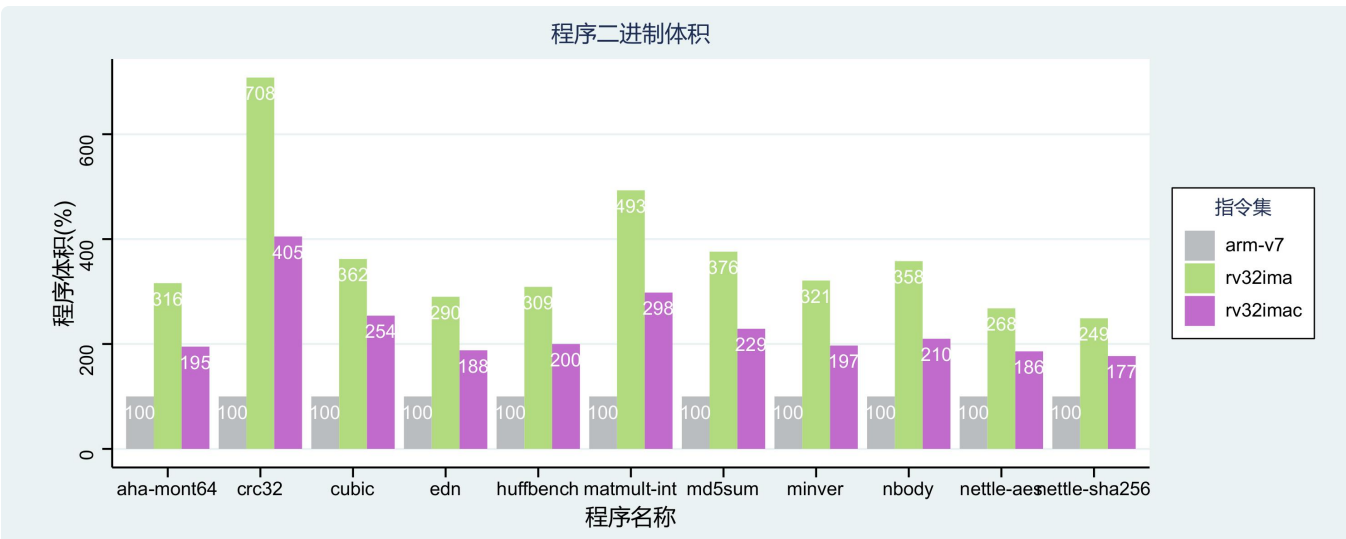


RISCV程序的二进制体积

- 二进制程序体积偏大

  复杂操作 -> 简单指令的组合

  压缩指令集（C扩展）



RISCV程序的二进制体积

- 二进制程序体积偏大

  复杂操作 -> 简单指令的组合

  压缩指令集（C扩展）

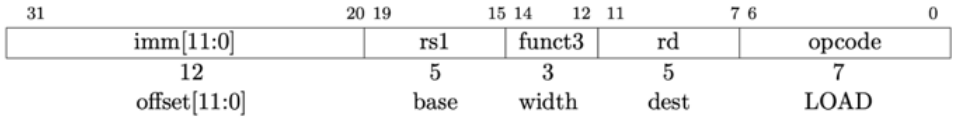
  新的压缩指令扩展 - Zce扩展

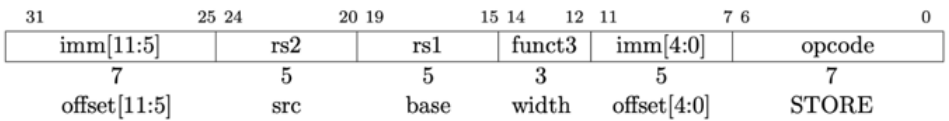
  - Zc: C扩展的子扩展
  - e: embedded，为嵌入式或小ram开发版准备的

字加载指令的优化

- 字加载指令： LW/SW，LD/SD(64 bit)，LH/SH，LB/SB ...

```
lui a0, 0
lw a1, 1024(a0)
```

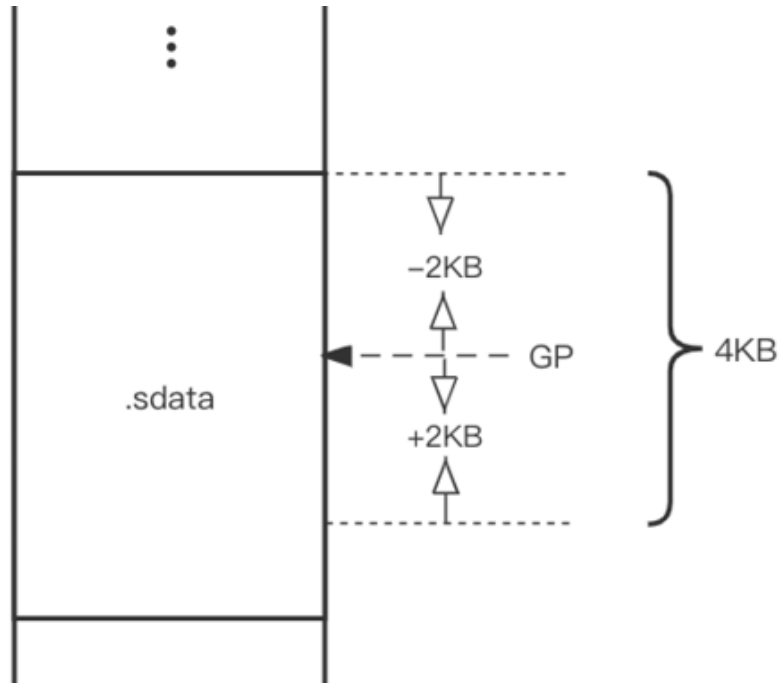




字加载指令的优化

- GP全局指针寄存器
  - __global_pointer$
  - defaul：.sdata + 0x800

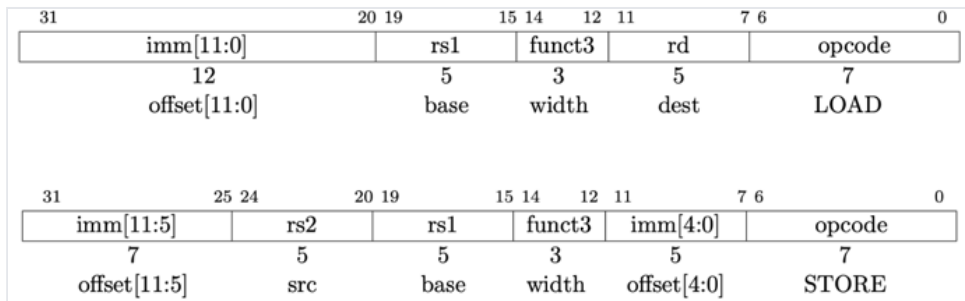
```
lw a1, 1024(gp)
```



字加载指令的优化 - LSGP

- LWGP/SWGP，LDGP/SDGP
- 约定GP为基址寄存器
- offset：±32KB

31:29	28:25	24:20	19:15	14:12	11:7	6 : 0	instruction
000	imm[8:2,10:9]		imm[15:11]	011	rd	0000111	LWGP



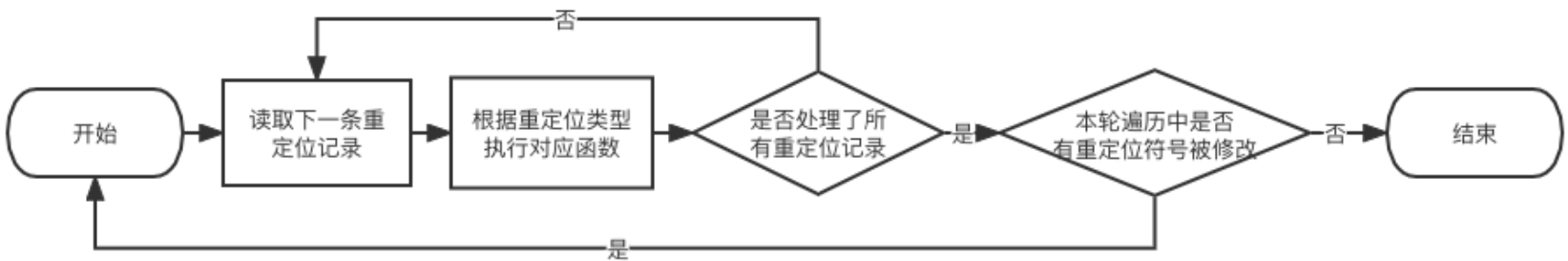
LLD链接器中的优化 - LSGP

```
lui a0, 0
lw a1, 1024(a0)
```

=>

```
lwgp a1, 1024
```

- RISCV架构的链接器松弛：[WIP][LLD][RISCV] Linker Relaxation



LLD链接器中的优化 - LSGP

```
lui a0, 0
lw a1, 1024(a0)
```

=>

```
lwgp a1, 1024
```

- RISCV架构的链接器松弛：[WIP][LLD][RISCV] Linker Relaxation
- 核心逻辑伪代码（部分）

```
for each rel in relocations
  if( rel.type is R_RISCV_HI20 and rel.inst is LUI)
    if( rel.nextInst is one of (LW ro LD or SW or SD) )
      if( rel.inst.offset is in range of Uint16 and aligned by 4 )
        Call removeInst(rel)
  else if ( rel.type is R_RISCV_LO12_I or R_RISCV_LO12_S)
    if( rel.inst is LW )
      if( rel.inst.offset is in range of Uint16 and aligned by 4 )
        Call repleaseInstByLWGP(rel)
        rel.type = R_RISCV_LWGP
    else if( rel.inst is SW )
      if( rel.inst.offset is in range of Uint16 and aligned by 4 )
        Call repleaseInstBySWGP(rel)
        rel.type = R_RISCV_SWGP
end
```

LLD链接器中的优化 - LSGP

```
lui a0, 0
lw a1, 1024(a0)
```

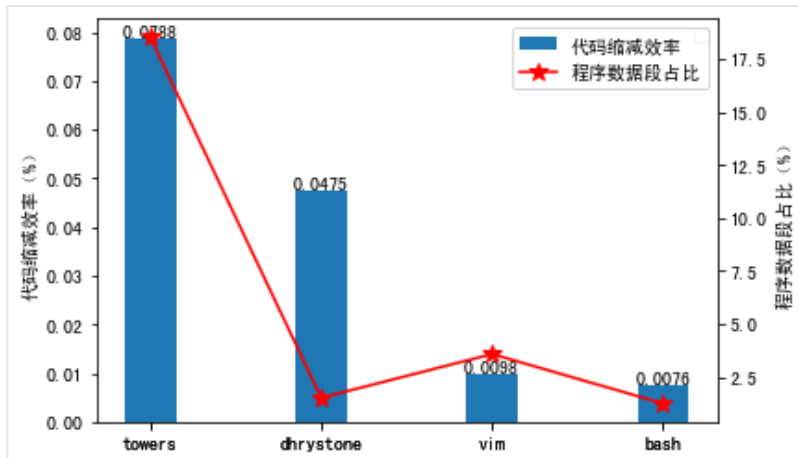
=>

```
lwgp a1, 1024
```

- RISCV架构的链接器松弛：[WIP][LLD][RISCV] Linker Relaxation
- 核心逻辑伪代码（部分）
- 边界情况
  1. c.lui
  2. lui指令的其他使用情况`int a = 0xfffff`
  3. a0被其他指令使用

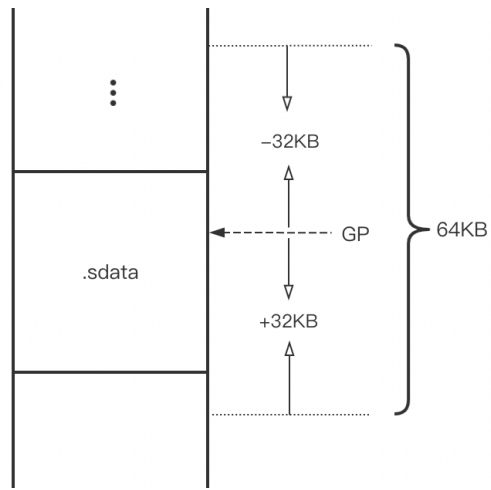
优化效率 - LSGP

- 与数据段大小，全局变量使用情况呈正相关
- LSGP在基准测试中的平均优化效率：0.06%
- Zce中其他指令优化效率：0.24% ～ 0.02%



LSGP存在的问题

- LSGP立即数设计不合理
- 部分LSGP的寻址能力被浪费
- 局限于优化.sdata段而忽略了其他可以被优化的数据段



总结

- LSGP 的优化原理
  - 扩大寻址能力
  - 替换指令组合
- 特殊情况
  - c.lui
  - lui的其他使用场景
  - 对其他指令的影响
- 优化效率
  - 0.06%
  - Zce指令每组指令优化效率： 0.02%～0.24%
- 存在的问题
  - 立即数设计
  - LSGP寻址能力

Reference

- riscv-non-isa/riscv-elf-psabi-doc GitHub.
- .sdata2/.sbss2 section generated by toolchain
- gcc gp (global pointer) register
- The dark side of RISC-V linker relaxation

Thanks