RISC-V二进制代码密度分析

刘曜

主要内容

- 引言
- 代码密度影响因素
- RISC-V开源编译器二进制代码密度对比分析
- 下一步工作

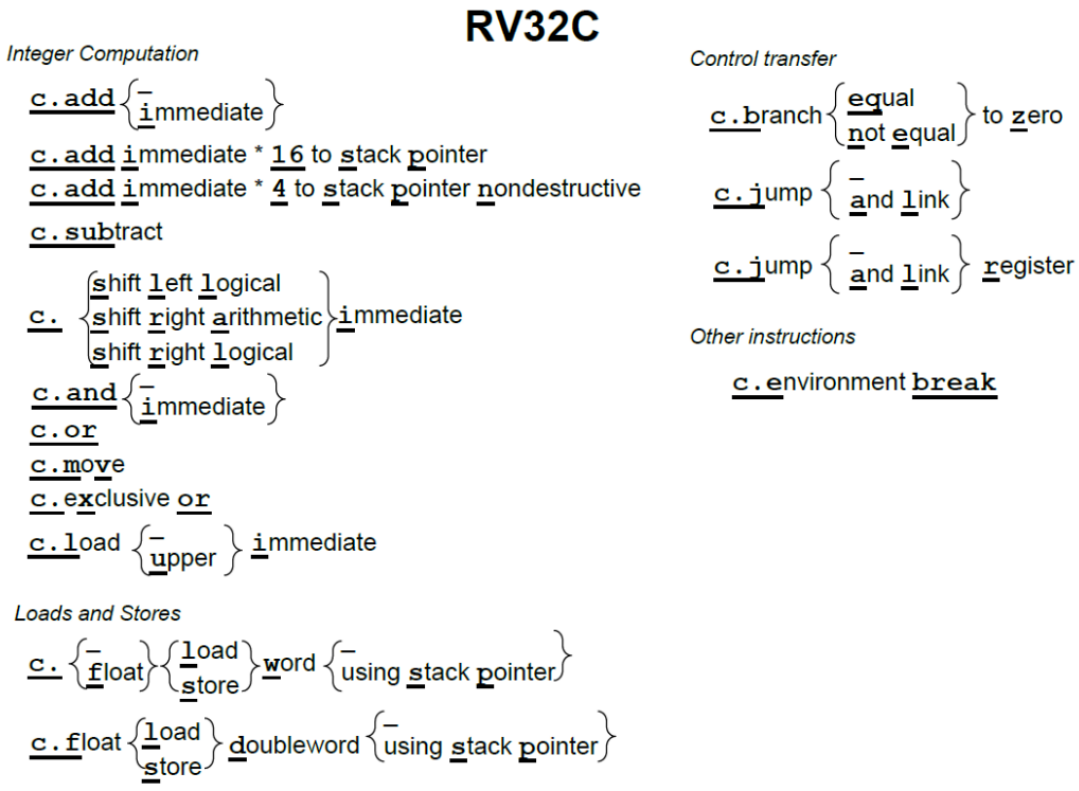
1.引言

- 开放指令架构 RISC-V 得到工业界和学术界的广泛关注
- AIoT（人工智能+物联网）时代，RISC-V的机遇和挑战
- 低功耗嵌入式CPU对代码密度的需求

2.代码密度影响因素

- 高密度的指令集架构——RV32C
- 软件工具链优化——RISC-V编译器优化、汇编器优化、链接器优化

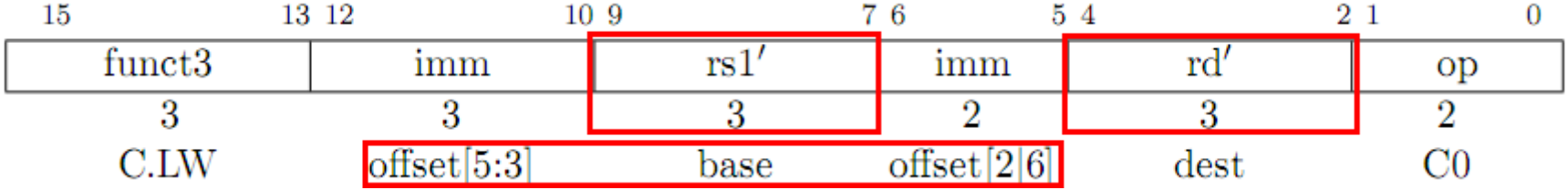
2.1RISC-V压缩指令集



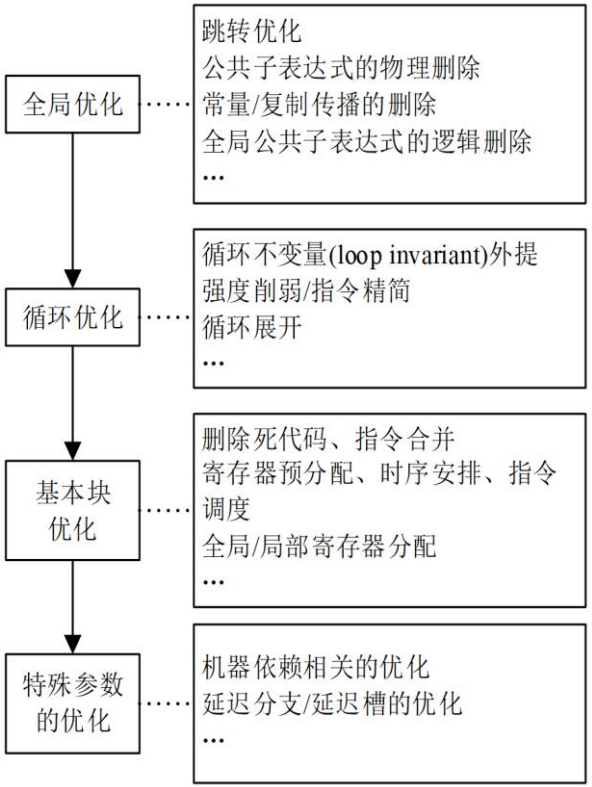
- 访问频率非常高的十个寄存器（s0-s1,a0-a5,sp,ra）
- 源操作数和目的操作数相同
- 立即数位数很小

2.2压缩指令的局限性

- 部分16位指令编码只用3位来表示8个寄存器（x8-x15）
- 立即数能表示的范围比32位指令小很多



2.3编译器优化技术


全局优化
跳转优化
公共子表达式的物理删除
常量/复制传播的删除
全局公共子表达式的逻辑删除
…
循环优化
循环不变量(loop invariant)外提
强度削弱/指令精简
循环展开
…
基本块优化
删除死代码、指令合并
寄存器预分配、时序安排、指令调度
全局/局部寄存器分配
…
特殊参数的优化
机器依赖相关的优化
延迟分支/延迟槽的优化
…

3.RISC-V编译器二进制代码密度分析

- 分析gcc-7.3.0和llvm-7.0优化后的二进制代码
- 跳转优化、循环优化以及汇编器的压缩指令生成

3.1跳转优化

```
int foo(int a)
{
    if(a == 0)
        return 10;
    else if(a == 1)
        return 100;
    else
        return 1000;
}
```

llvm 编译器

```
00000000 <foo>:
0:   85aa          mv    a1,a0
2:   4505          li    a0,1
4:   00a58663      beq   a1,a0,10 <fooa+0x10>
8:   3e800513      li    a0,1000
c:   c589          beqz  a1,16 <fooa+0x16>
e:   a029          j     18 <fooa+0x18>
10: 06400513       li    a0,100
14: e191           bnez  a1,18 <fooa+0x18>
16: 4529           li    a0,10
18: 8082           ret
```

gcc 编译器

```
00000000 <foo>:
0: 47a9            li    a5,10
2: c901            beqz  a0,12 <.L1>
4: 4705            li    a4,1
6: 06400793        li    a5,100
a: 00e50463        beq   a0,a4,12 <.L1>
e: 3e800793        li    a5,1000
00000012 <.L1>:
12: 853e           mv    a0,a5
14: 8082           ret
```

3.2循环优化

```
void foo(int *a,int *b)
{
    int i;
    for(i=0; i<10; i++)
        a[i]=b[i];
}
```

llvm 编译器

```
00000000 <foo>:
0:   02800613   li     a2,40
4:   4681       li     a3,0
6:   00d50733   add    a4,a0,a3
a:   00d587b3   add    a5,a1,a3
e:   439c       lw     a5,0(a5)
10:  c31c       sw     a5,0(a4)
12:  0691       addi   a3,a3,4
14:  fec699e3   bne    a3,a2,6 <fooa+0x6>
18:  8082       ret
```

gcc 编译器

```
00000000 <foo>:
0:  02858793  addi     a5,a1,40

00000004 <.L2>:
4:  4198      lw       a4,0(a1)
6:  0591      addi     a1,a1,4
8:  0511      addi     a0,a0,4
a:  fee52e23   sw      a4,-4(a0)
e:  fef59be3  bne      a1,a5,4 <.L2>
12:8082                ret
```

3.3汇编器优化分析

addi rd,zero,imm
add rd,zero,rs2
这两条在汇编过程中可以生成16位压缩指令编码格式

汇编测试代码	llvm 汇编器	gnu 汇编器
foo:	00000000 <foo>:	00000000 <foo>:
addi a2, zero, 1	0:4605 li a2,1	0:00100613 li a2,1
add a0, zero, a1	2:852e mv a0,a1	4:00b00533 add a0,zero,a1
ret	4:8082 ret	8:8082 ret

3.4链接器优化

```
00000000 <main>:
 0: ff010113   addi   sp,sp,-16
 4: 00112623   sw     ra,12(sp)
 8: 00000537   lui    a0,0x0
 c: 00050513   mv     a0,a0
10: 000005b7   lui    a1,0x0
14: 00058593   mv     a1,a1
18: 00000097   auipc  ra,0x0
1c: 000080e7   jalr   ra
20: 00c12083   lw     ra,12(sp)
24: 01010113   addi   sp,sp,16
28: 00000513   li     a0,0
2c: 00008067   ret
```

```
000101b0 <main>:
   101b0: ff010113 addi sp,sp,-16
   101b4: 00112623 sw   ra,12(sp)
   101b8: 00021537 lui  a0,0x21
   101bc: a1050513 addi a0,a0,-1520 # 20a10 <string1>
   101c0: 000215b7 lui  a1,0x21
   101c4: a1c58593 addi a1,a1,-1508 # 20a1c <string2>
   101c8: 288000ef jal  ra,10450 <printf>
   101cc: 00c12083 lw   ra,12(sp)
   101d0: 01010113 addi sp,sp,16
   101d4: 00000513 li   a0,0
   101d8: 00008067 ret
```

4.下一步工作

- 实现在LLVM编译器上对代码密度优化效率的提高
- 实现在GCC编译器上对代码密度优化效率的提高
- 实现GNU汇编器对所有RISC-V压缩指令的覆盖生成

4.1团队与产品介绍

湖南卡姆派乐信息科技有限公司

- 公司核心团队成员来自国防科技大学技术团队，先后负责并完成多项CPU、DSP、AI芯片、RISC-V芯片等集成开发环境、编译器、操作系统等软件的设计与研发工作
- 为芯片厂家提供单核、多核、异构多核芯片集成开发环境、编译器、调试器、高性能库、操作系统等定制服务
- 2019年10月18日在浙江乌镇举行的“世界互联网大会”上发布了国内首款自研RISC-V集成开发环境---卡姆派乐IDE（COMPILER IDE）
- 2019年11月7日，正式支持兆易创新RISC-V GD32VF103芯片

4.2卡姆派乐IDE

湖南卡姆派乐信息科技有限公司

- 简洁易用：在启动、开发、调试阶段无处不追求简洁易用
- 代码密度高：编译器、库函数、驱动优化，满足MCU对代码密度的要求
- 稳定性高：相较于eclipse版本稳定性更高
- 可定制：编译器、IDE、操作系统、库函数、检测分析工具均可定制

4.3展望未来

湖南卡姆派乐信息科技有限公司

- 作为国内首家专业提供芯片系统软件解决方案的高新科技企业，卡姆派乐公司会一直致力于国产芯片开发环境的研发和推广工作，特别是RISC-V生态系统的建设和完善工作
- 希望跟业界相关伙伴能够更多合作，携手发展，也希望能够为国家“缺芯少魂”的现状贡献一点点绵薄之力

谢谢