奕斯伟计算RISC-V GCC工具链开发实践分享

高斐 王峰

北京奕斯伟计算技术股份有限公司

I. 奕斯伟计算 RISC-V GCC 工具链开发实践成果

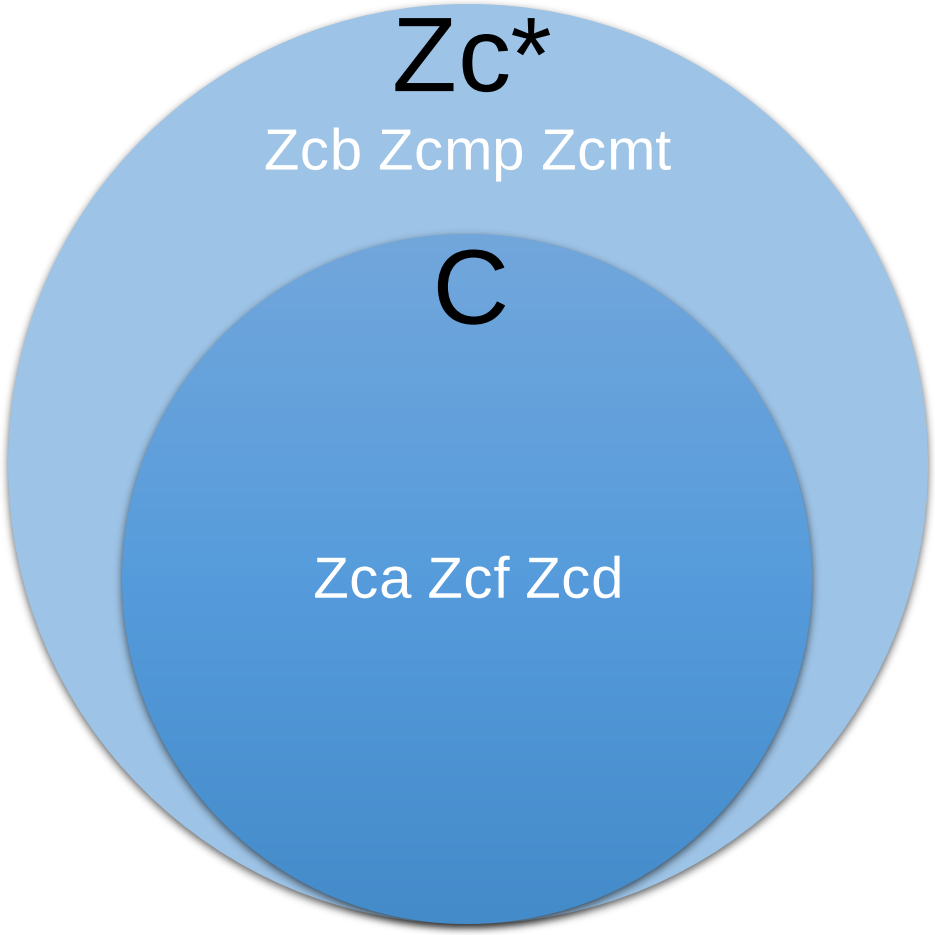
ESWIN

RISC-V 工具链是 RISC-V 软件生态的基础，负责将高级编程语言转换和优化成可以在 RISC-V 平台上运行的机器码。

奕斯伟计算工具链团队深度参与了 RISC-V GCC 工具链的生态建设，并获得下列成果：

maintainer	有4名成员获得 GCC maintainer write after approval 权限
RISC-V新扩展的贡献者	RISC-V GCC Zc*, Zicond, Crypto Vector, BF16 等新扩展的主要贡献者
130+ patches	超过130个补丁被 GCC、Binutils 等社区接收

II. Zc* 扩展 – 简介



Zc*：通过指令压缩来减少 code size 的一系列子扩展总称。

Zc*子扩展	压缩类型	主要实现模块
Zca	整型指令	汇编器
Zcf	单精度浮点指令	汇编器
Zcd	双精度浮点指令	汇编器
Zcb	整型指令	汇编器
Zcmp	堆栈操作指令	编译器
Zcmt	函数调用指令	链接器

II. Zc* 扩展 – Zcmp cm.push 入栈压缩效果

ESWIN

-march=rv32imafc

```
addi sp,sp,-96 ;#cm.push(1)
sw s2,80(sp)   ;#cm.push(2)
sw s3,76(sp)   ;#cm.push(3)
sw s4,72(sp)   ;#cm.push(4)
sw ra,92(sp)   ;#cm.push(5)
sw s0,88(sp)   ;#cm.push(6)
sw s1,84(sp)   ;#cm.push(7)
sw s5,68(sp)   ;#cm.push(8)
sw s6,64(sp)   ;#cm.push(9)
sw s7,60(sp)   ;#cm.push(10)
sw s8,56(sp)   ;#cm.push(11)
sw s9,52(sp)   ;#cm.push(12)
sw s10,48(sp) ;#cm.push(13)
sw s11,44(sp) ;#cm.push(14)
```

-march=rv32imaf_zcmp

```
cm.push {ra,s0-s11},-96
```

II. Zc* 扩展 – Zcmp cm.popret 出栈压缩效果

ESWIN

-march=rv32imafc

```
lw ra,92(sp)          ;#cm.popret(1)
lw s0,88(sp)          ;#cm.popret(2)
lw s1,84(sp)          ;#cm.popret(3)
lw s2,80(sp)          ;#cm.popret(4)
lw s3,76(sp)          ;#cm.popret(5)
lw s4,72(sp)          ;#cm.popret(6)
lw s5,68(sp)          ;#cm.popret(7)
lw s6,64(sp)          ;#cm.popret(8)
lw s7,60(sp)          ;#cm.popret(9)
lw s8,56(sp)          ;#cm.popret(10)
lw s9,52(sp)          ;#cm.popret(11)
lw s10,48(sp)         ;#cm.popret(12)
lw s11,44(sp)         ;#cm.popret(13)
addi sp,sp,96         ;#cm.popret(14)
ret                   ;#cm.popret(15)
```

Committed to GCC14

-march=rv32imaf_zcmp

```
cm.popret {ra,s0-s11}, 96
```

II. Zc* 扩展– Zcmp {a0, a1}和s寄存器相互搬移压缩效果

ESWIN

cm.mvsa01

-march=rv32imafc

```
mv      s3,a0
mv      s0,a1
```

-march=rv32imaf_zcmp

```
cm.mvsa01      s3,s0
```

cm.mva01s

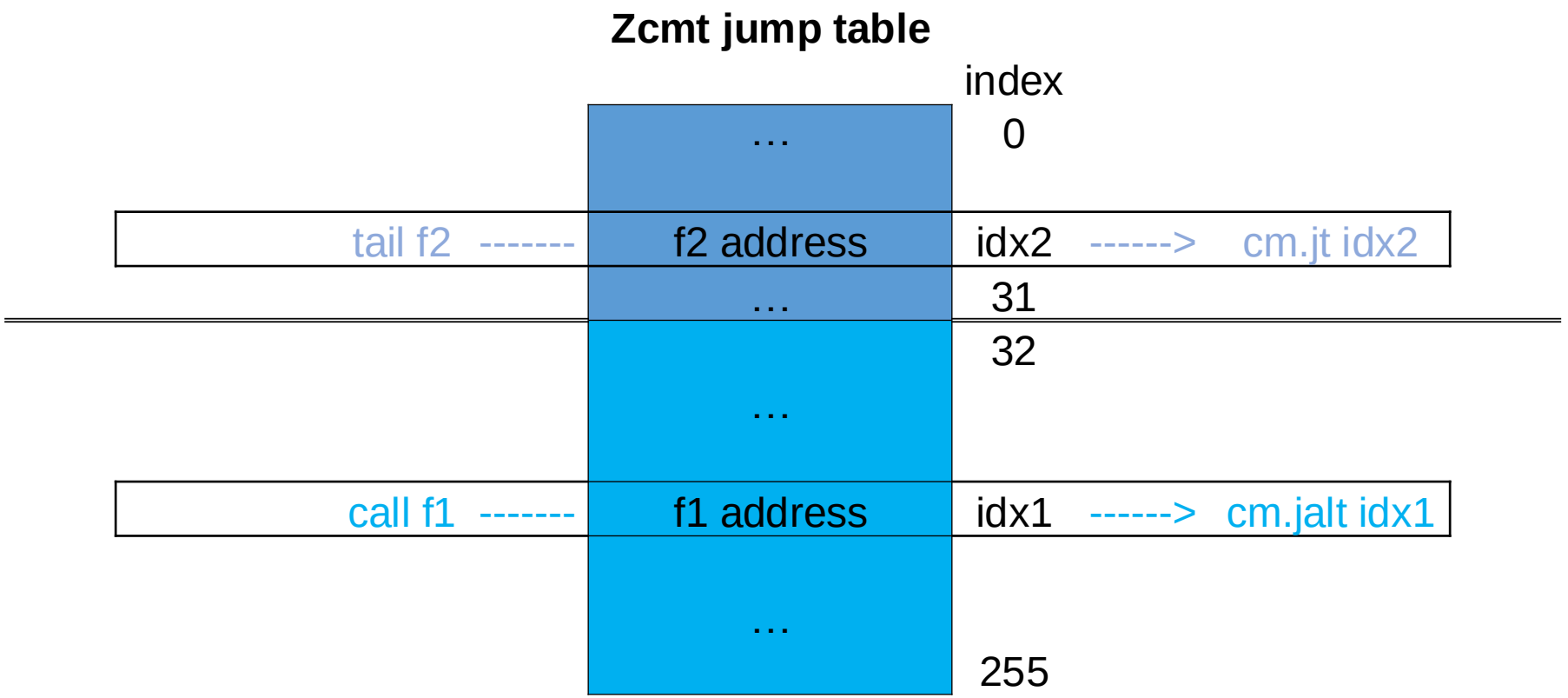
-march=rv32imafc

```
mv      a1,s1
mv      a0,s0
```

-march=rv32imaf_zcmp

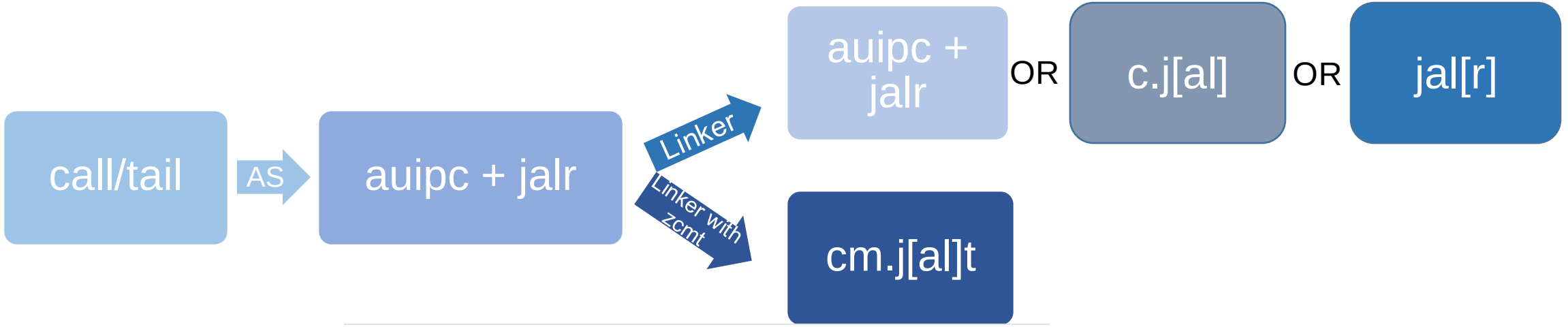
```
cm.mva01s      s0,s1
```

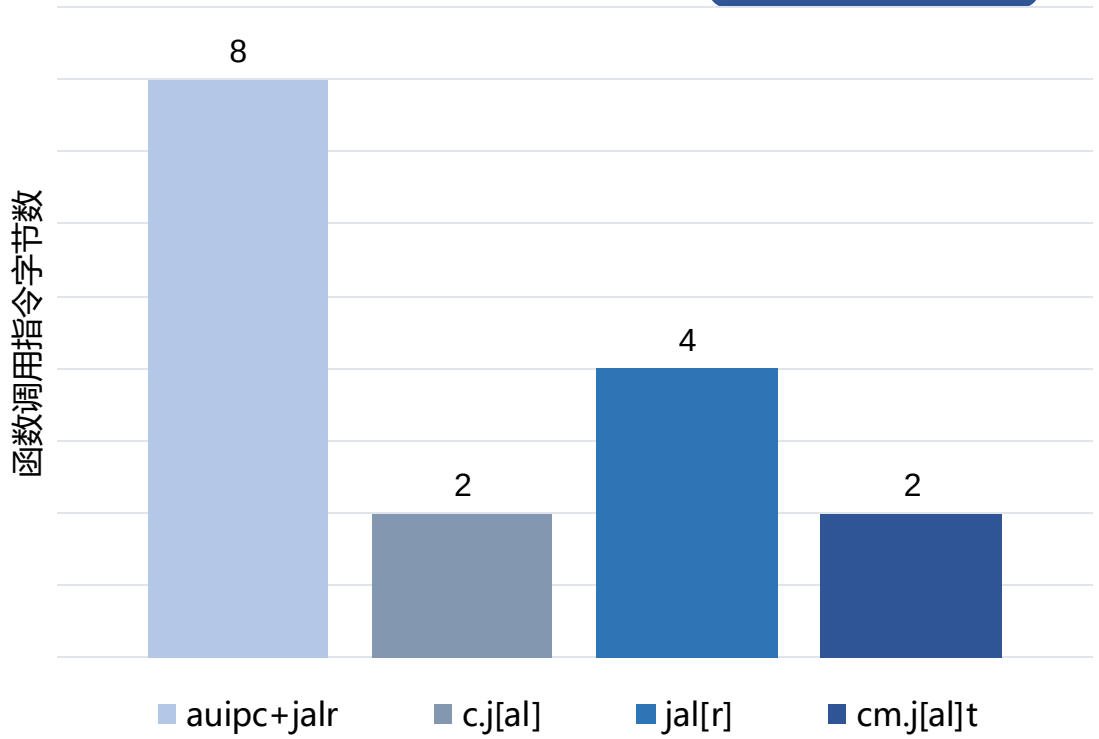
II. Zc* 扩展 – Zcmt 优化原理



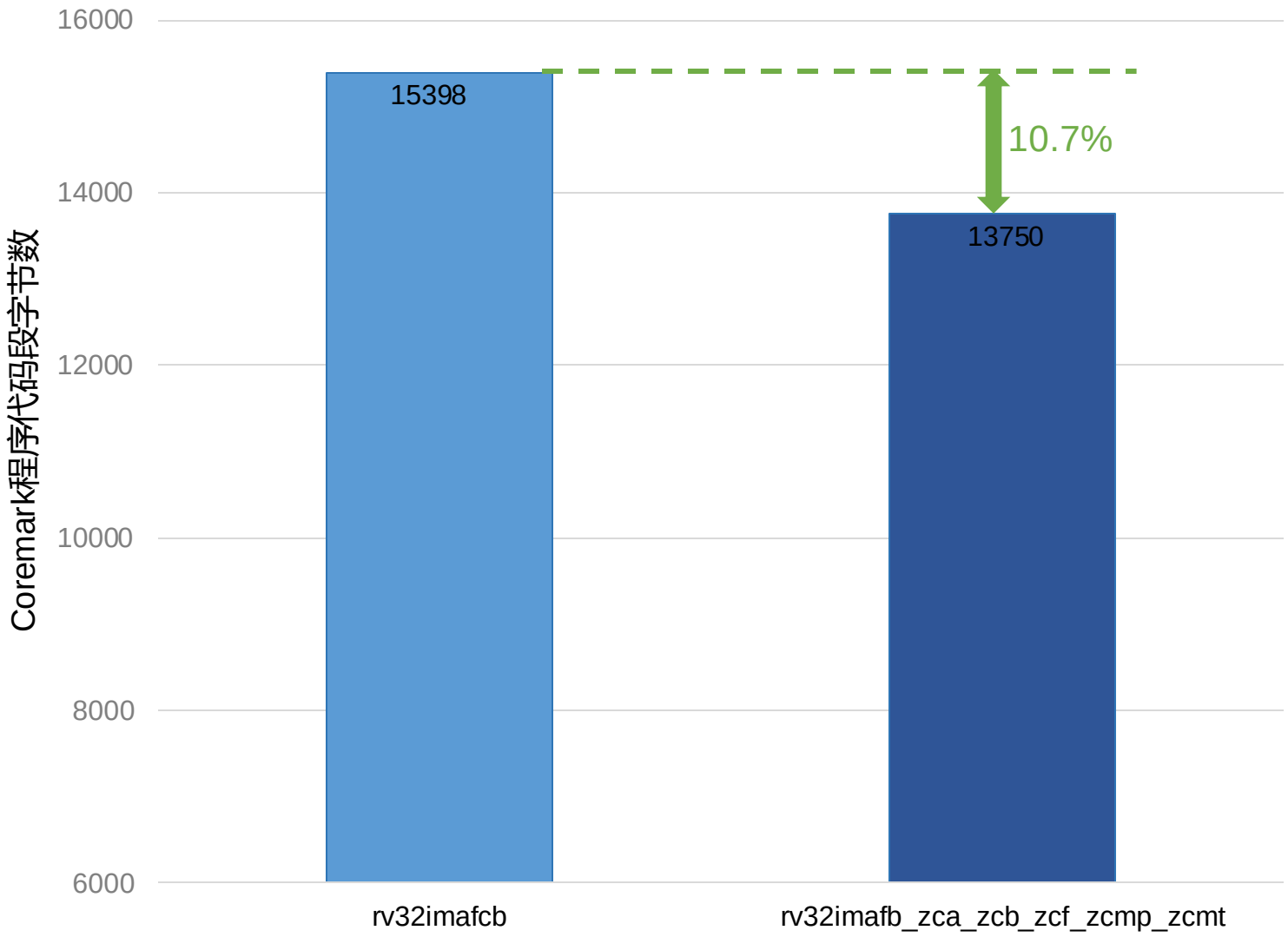
II. Zc* 扩展 – Zcmt 优化效果

ESWIN





II. Zc* 扩展 – Zc* Coremark code size 减少效果



II. Zc* 扩展 – GCC 社区状态

ESWIN

- GCC14工具链已经支持 Zca, Zcb, Zcf, Zcd 和 Zcmp。
- Zcmt 正在逐步合入主线中。
- GCC 社区 Zc* 扩展主要贡献者：奕斯伟计算和 PLCT。

III. Zicond 扩展 – 简介

ESWIN

- Zicond 定义了两条 czero 指令，**通过编译器的优化可以消除条件分支跳转指令**，避免分支预测失败带来的代价。
- Zicond **硬件实现更简单**，相对于 SFB 和 conditional move。

czero.eqz rd, rs1, rs2

语义
rd = (rs2 == 0) ? 0 : rs1

czero.nez rd, rs1, rs2

语义
rd = (rs2 != 0) ? 0 : rs1

III. Zicond 扩展 – 消除条件分支示例1

ESWIN

```
long foo (long x, long y, long c)
{
   return c ? x : x + y;
}
```

-march=rv64gc

```
foo:
        bne     a2,zero,.L2
        add     a0,a0,a1
.L2:
        ret
```

-march=rv64gc_zicond

```
foo:
        czero.nez  a2,a1,a2
        add        a0,a0,a2
        ret
```

III. Zicond 扩展 – 消除条件分支示例2

ESWIN

```
long bar (long x, long y, long c)
{
  return c ? x : y;
}
```

-march=rv64gc

```
bar:
        beq     a2,zero,.L5
        mv      a1,a0
.L5:
        mv      a0,a1
        ret
```

-march=rv64gc_zicond

```
bar:
        czero.nez       a1,a1,a2
        czero.eqz       a0,a0,a2
        add     a0,a0,a1
        ret
```

III. Zicond 扩展 – spec 示例

ESWIN

Operation	Instruction sequence	Length
Conditional add, if zero rd = (rc == 0) ? (rs1 + rs2) : rs1	czero.nez rd, rs2, rc add rd, rs1, rd	2 insns
Conditional add, if non-zero rd = (rc != 0) ? (rs1 + rs2) : rs1	czero.eqz rd, rs2, rc add rd, rs1, rd	
Conditional subtract, if zero rd = (rc == 0) ? (rs1 - rs2) : rs1	czero.nez rd, rs2, rc sub rd, rs1, rd	
Conditional subtract, if non-zero rd = (rc != 0) ? (rs1 - rs2) : rs1	czero.eqz rd, rs2, rc sub rd, rs1, rd	
Conditional bitwise-or, if zero rd = (rc == 0) ? (rs1 | rs2) : rs1	czero.nez rd, rs2, rc or rd, rs1, rd	
Conditional bitwise-or, if non-zero rd = (rc != 0) ? (rs1 | rs2) : rs1	czero.eqz rd, rs2, rc or rd, rs1, rd	
Conditional bitwise-xor, if zero rd = (rc == 0) ? (rs1 ^ rs2) : rs1	czero.nez rd, rs2, rc xor rd, rs1, rd	
Conditional bitwise-xor, if non-zero rd = (rc != 0) ? (rs1 ^ rs2) : rs1	czero.eqz rd, rs2, rc xor rd, rs1, rd	
Conditional bitwise-and, if zero rd = (rc == 0) ? (rs1 & rs2) : rs1	and rd, rs1, rs2 czero.eqz rtmp, rs1, rc or rd, rd, rtmp	3 insns (requires 1 temporary)
Conditional bitwise-and, if non-zero rd = (rc != 0) ? (rs1 & rs2) : rs1	and rd, rs1, rs2 czero.nez rtmp, rs1, rc or rd, rd, rtmp	
Conditional select, if zero rd = (rc == 0) ? rs1 : rs2	czero.nez rd, rs1, rc czero.eqz rtmp, rs2, rc or rd, rd, rtmp	
Conditional select, if non-zero rd = (rc != 0) ? rs1 : rs2	czero.eqz rd, rs1, rc czero.nez rtmp, rs2, rc or rd, rd, rtmp	

Zicond 扩展 – 复杂场景下的条件分支消除

ESWIN

```
unsigned long foo (unsigned long x,
                   unsigned long y,
                   unsigned long c)
{
  unsigned long ret;
  if (c != 10)
    ret = (x*x + y*y ) + 2 * (x + y);
  else
    ret = x - y;
  return ret;
}
```

-march=rv64gc

```
foo:
        li      a5,10
        beq     a2,a5,.L2
        mul     a5,a1,a1
        mul     a4,a0,a0
        add     a5,a5,a4
        add     a0,a0,a1
        slli    a0,a0,1
        add     a0,a5,a0
        ret
.L2:
        sub     a0,a0,a1
        ret
```

-march=rv64gc_zicond

```
foo:
        mul     a5,a1,a1
        mul     a4,a0,a0
        add     a5,a5,a4
        add     a4,a0,a1
        slli    a4,a4,1
        add     a5,a5,a4
        sub     a0,a0,a1
        addi    a2,a2,-10
        czero.nez a0,a0,a2
        czero.eqz a2,a5,a2
        add     a0,a2,a0
        ret
```

III. Zicond 扩展 – GCC 社区状态

ESWIN

- GCC14 工具链已经支持 Zicond 基本功能和优化。
- 奕斯伟计算正在优化更复杂的场景 (SUBREG, ZERO_EXTEND, SIGN_EXTEND, …) , 期待在GCC15 发布。
- GCC 社区 Zicond 扩展主要贡献者：奕斯伟计算和 Ventana。

IV. Vector Crypto 扩展 – 简介

ESWIN

- Vector Crypto 扩展的目标:
  - 增加硬件的加密能力。
  - 提高加密计算的性能和安全性。
- Vector Crypto 包括的子扩展:
  - Zvkb: Vector Cryptography Bit-manipulation.
  - Zvkg: Efficient implementation of GHASHH.
  - Zvkned: Accelerating procession of AES block cipher.
  - Zvknh[ab]: Accelerating SHA-2.
  - Zvksed: Accelerating functions of SM4 block cipher.
  - Zvksh: Accelerating functions of SM3 Hash Function.

IV. Vector Crypto 扩展 – Intrinsic 接口实现

ESWIN

Vector Crypto intrinsic 已合入 GCC14:

```
vuint8mf8_t __riscv_vandn_vv_u8mf8(vuint8mf8_t vs2, vuint8mf8_t vs1, size_t vl);

vuint8mf8_t __riscv_vandn_vx_u8mf8(vuint8mf8_t vs2, uint8_t rs1, size_t vl);

vuint64m2_t __riscv_vclmul_vx_u64m2(vuint64m2_t vs2, uint64_t rs1, size_t vl);

vuint64m4_t __riscv_vclmul_vv_u64m4(vuint64m4_t vs2, vuint64m4_t vs1, size_t vl);

vuint32m1_t __riscv_vgmul_vv_u32m1(vuint32m1_t vd, vuint32m1_t vs2, size_t vl);

vuint32m2_t __riscv_vgmul_vv_u32m2(vuint32m2_t vd, vuint32m2_t vs2, size_t vl);

……
```

rvv-intrinsic-doc/auto-generated/vector-crypto/intrinsic_funcs at main · riscv-non-isa/rvv-intrinsic-doc · GitHub

IV. Vector Crypto 扩展 – 指令合并优化效果

ESWIN

```
#include <stdint.h>
void circular_left_shift(uint32_t *dest, uint32_t *input, uint32_t
*shift)
{
  for (int i = 0; i < n; i++)
    dest[i] = (input[i] << shift[i]) | (input[i] >> (32 – shift[i]));
}
```

gcc-master: -march=rv32gcv

```
…
vsrl.vv    v1,v2,v1
vsll.vv    v2,v2,v3
vor.vv     v1,v1,v2
…
```

Committed to gcc-master

gcc-master: -march=rv32gcv_zvbb

```
…
vrol.vv    v1,v1,v2
…
```

IV. Vector Crypto 扩展 – GCC社区状态

ESWIN

- 奕斯伟计算贡献的 Vector Crypto intrinsic 接口已经合入到 GCC14。
- 奕斯伟计算正在向 GCC 社区贡献 Vector Crypto 指令合并优化。
- GCC 社区 Vector Crypto 扩展主要贡献者: 奕斯伟计算。

V. BF16 扩展 – 简介

ESWIN

- BF16 扩展目的:
  - 加速 AI 应用。
  - 显著减少内存占用，在维持计算精度的同时提高计算吞吐率。
- BF16 包括的子扩展:
  - Zfbfmin: 标量 BF16 最小指令集。
  - Zvfbfmin: 向量 BF16 最小指令集。
  - Zvfbfwma: 向量 BF16 Widening mul-add 指令。

V. BF16 扩展 – 增加GCC对 RISC-V BF16 类型的支持

ESWIN

GCC14

gcc-master

```
bf16.c:1:8: error: unknown type name '__bf16'
    1 | extern __bf16 bf;
      |        ^~~~~~
```

Committed to gcc-master

V. BF16 扩展 – GCC自动向量化优化效果

ESWIN

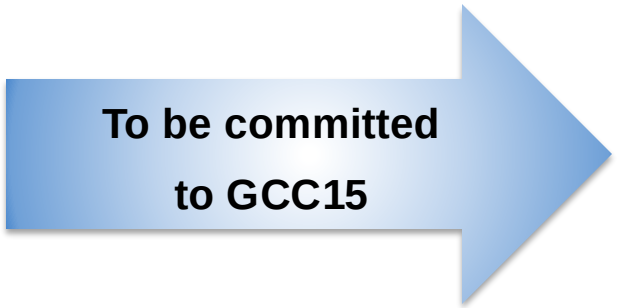
```
void vfncvt_float_BFloat16 (__bf16 *dst, float *a, int
n)
{
    for (int i = 0; i < n; i++)
      dst[i] = (__bf16)a[i];
}
```

gcc-master: -march=rv32gcv

```
.L3:    flw       fa0,0(s0)
        …
        call      __truncsfbf2

        fmv.x.s   a5,fa0
        …
        sh        a5,-2(s1)

        bne       s2,s0,.L3
```



ESWIN Computing GCC: -march=rv32gcv_zvfbfmin

```
.L3:    vsetvli   a5,a2,e8,mf4,ta,ma

        vle32.v   v1,0(a1)
        …
        vfncvtbf16.f.f.w    v1,v1
        …
        vse16.v   v1,0(a0)
        …
        bne       a2,zero,.L3
```

V. BF16 扩展 – GCC自动向量化优化效果

ESWIN

```
void vwmacc_float_bf16 (float *__restrict dst,  __bf16 *__restrict a,
__bf16 *__restrict b, int n)
{
    for (int i = 0; i < n; i++)
      dst[i] += (float) (a[i] * b[i]);
}
```

gcc-master: -march=rv32gcv_zfbfmin

```
.L3:
        flh         fa5,0(a1)
        flh         fa4,0(a2)
        flw         fa3,0(a0)
        fcvt.s.bf16         fa5,fa5
        fcvt.s.bf16         fa4,fa4
        …
        fmadd.s  fa5,fa5,fa4,fa3
        …
        fsw         fa5,-4(a0)
        bne         a5,a0,.L3
```

To be committed to GCC15

ESWIN Computing GCC: -march=rv32gcv_zvfbfwma

```
.L3:
        vsetvli     a5,a3,e16,mf2,tu,ma

        vle32.v   v4,0(a0)
        vle16.v   v2,0(a1)
        vle16.v   v3,0(a2)
        …
        vfwmaccbf16.vv   v1,v3,v2
        vse32.v  v1,0(a4)
        …
        bne         a3,zero,.L3
```

V. BF16 扩展 – GCC社区状态

ESWIN

- 奕斯伟计算贡献的 RISC-V BF16 数据类型支持、 Zfbfmin、Zvfbfmin 和 Zvfbfwma 子扩展的 intrinsic 接口已合入 gcc-master，将于 GCC15 正式发布。
- 奕斯伟计算正在向 GCC 社区贡献 RISC-V BF16 自动向量化优化以及更多的 RISC-V BF16 libgcc 支持。
- GCC 社区 RISC-V BF16 扩展主要贡献者: 奕斯伟计算。

I. 奕斯伟计算 RISC-V GCC 工具链开发实践成果

II. RISC-V Zc* 扩展实现与优化

III. RISC-V Zicond 扩展实现与优化

IV. RISC-V Vector Crypto 扩展实现与优化

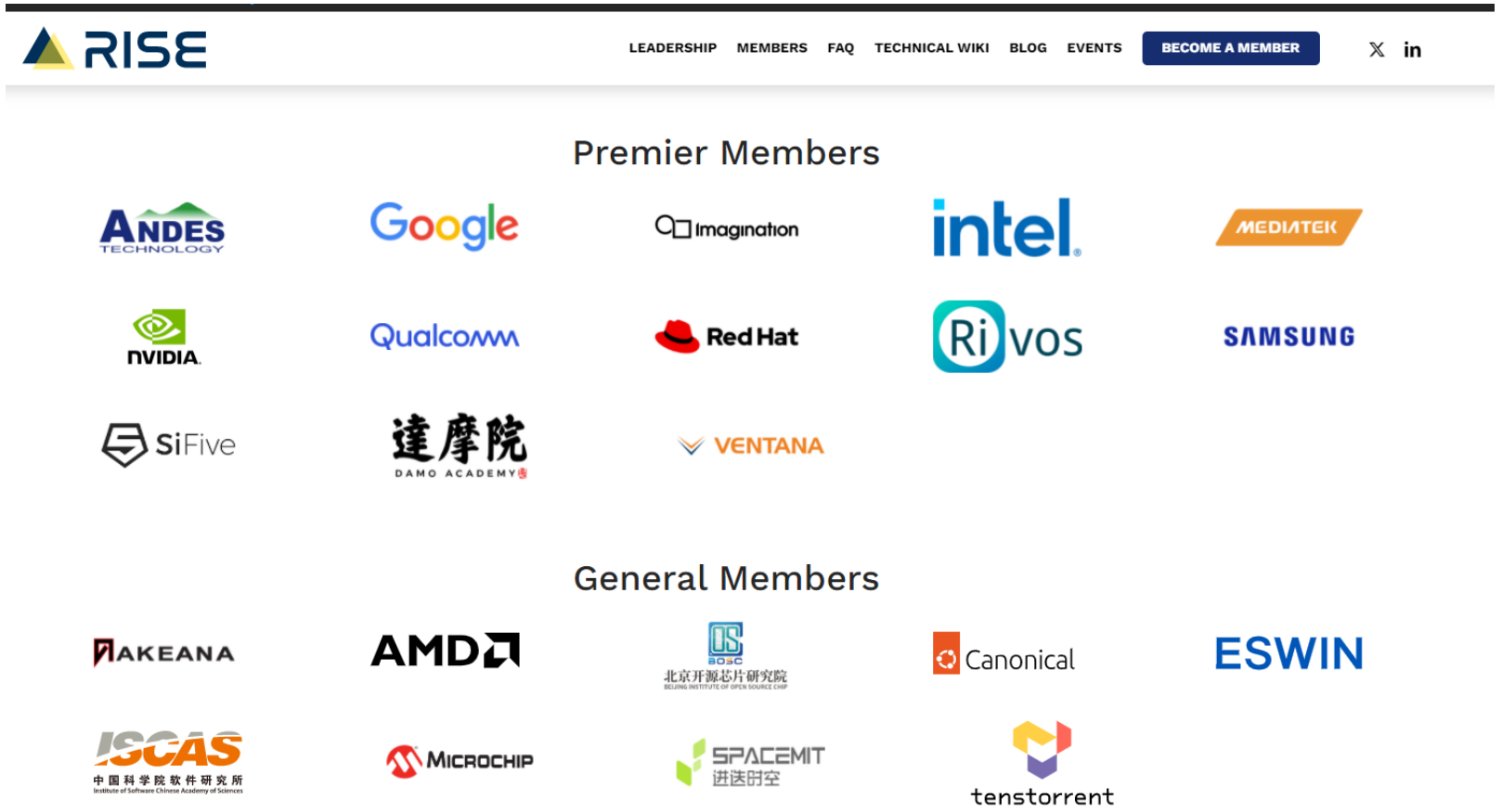
V. RISC-V BF16 扩展实现与优化

VI. 奕斯伟计算携手合作伙伴共建 RISC-V 开源软件生态

VI. 奕斯伟计算携手合作伙伴共建RISC-V开源软件生态

ESWIN

奕斯伟计算 于2024年加入 RISE，携手合作伙伴加速 RISC-V 开源软件生态建设，承担了 Compilers and Toolchains WG 的 RISC-V GCC 工具链相关工作。



ESWIN

Thanks

北京奕斯伟计算技术股份有限公司

BEIJING ESWIN COMPUTING TECHNOLOGY CO., LTD.

www.eswincomputing.com

奕斯伟计算官方公众号