2024 RISC-V Summit China

RVV Auto-Vectorization in GCC

Pan Li – Intel

Liu, Hongtao - Intel

Legal Notices and Disclaimers

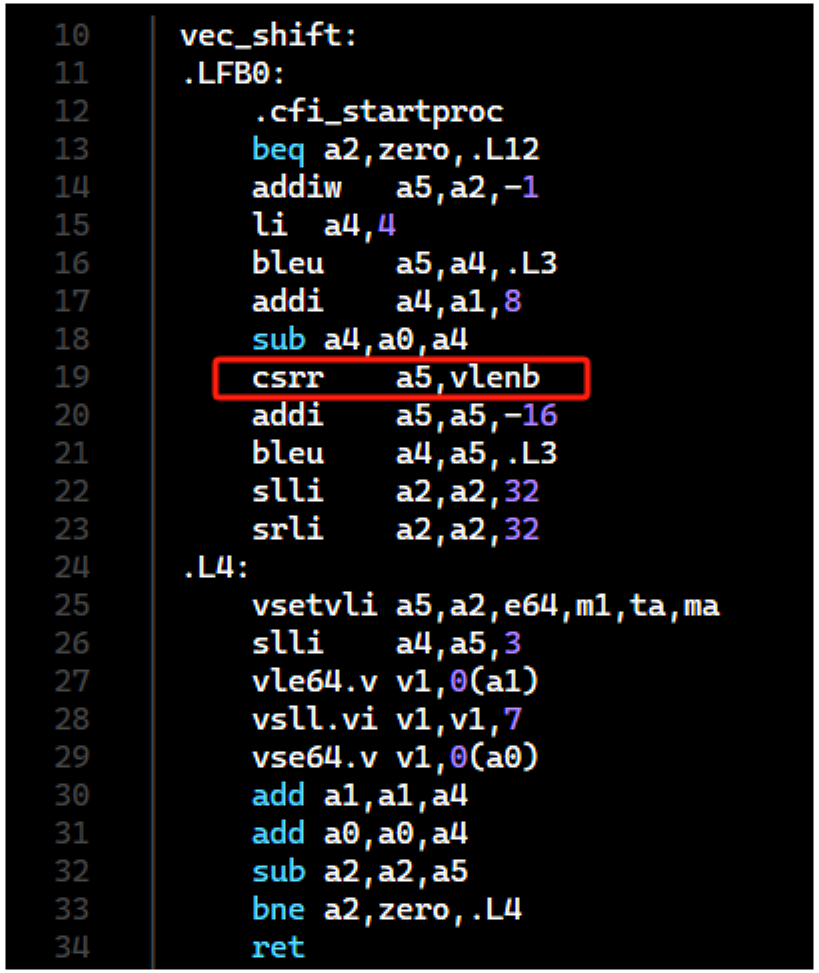
Agenda

Scalable (default) – by RiVAI, Ventana Micro and Intel

-mrvv-vector-bits=**scalable**

Any vlen is a multiple of zvl bits.

```
void
vec_shift (int64_t *out, int64_t *a,
           uint32_t size)
{
  for (uint32_t  i = 0; i < size; i++)
    out[i] = a[i] << 7;
}
```



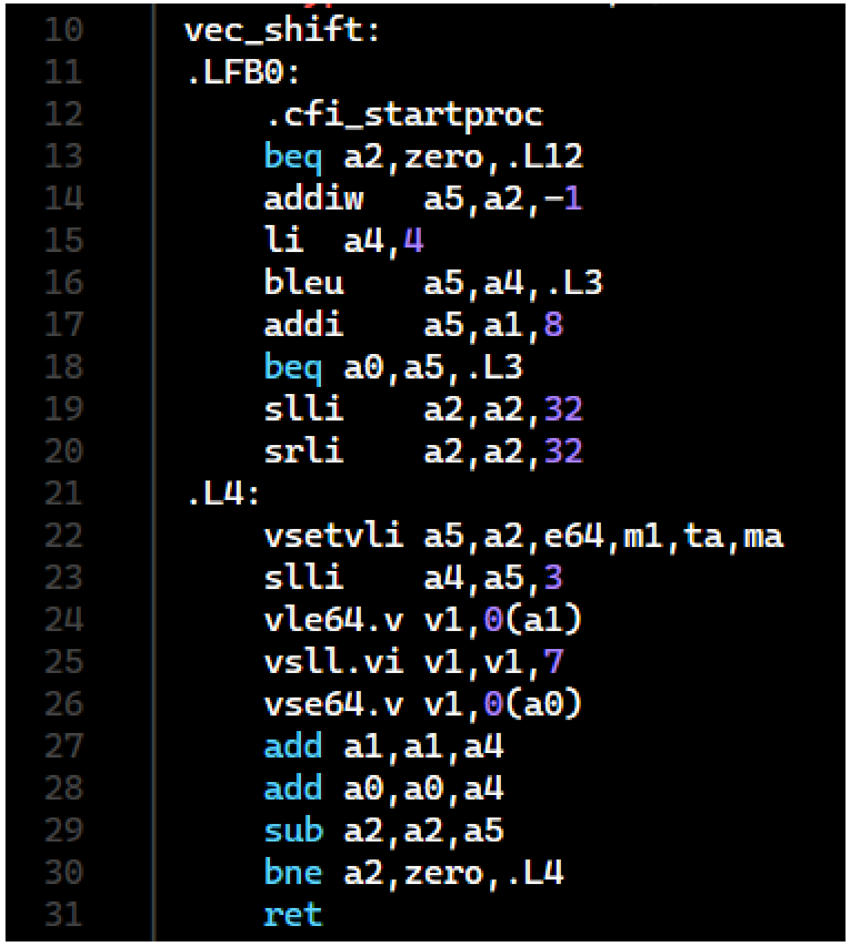
```
vec_shift:
.LFB0:
	.cfi_startproc
	beq a2,zero,.L12
	addiw	a5,a2,-1
	li  a4,4
	bleu	a5,a4,.L3
	addi	a4,a1,8
	sub a4,a0,a4
	csrr	a5,vlenb
	addi	a5,a5,-16
	bleu	a4,a5,.L3
	slli	a2,a2,32
	srli	a2,a2,32
.L4:
	vsetvli a5,a2,e64,m1,ta,ma
	slli	a4,a5,3
	vle64.v v1,0(a1)
	vsll.vi v1,v1,7
	vse64.v v1,0(a0)
	add a1,a1,a4
	add a0,a0,a4
	sub a2,a2,a5
	bne a2,zero,.L4
	ret
```

Fixed - by RiVAI, Ventana Micro and Intel

-mrvv-vector-bits=**zvl**

The vlen is exactly same as zvl bits.

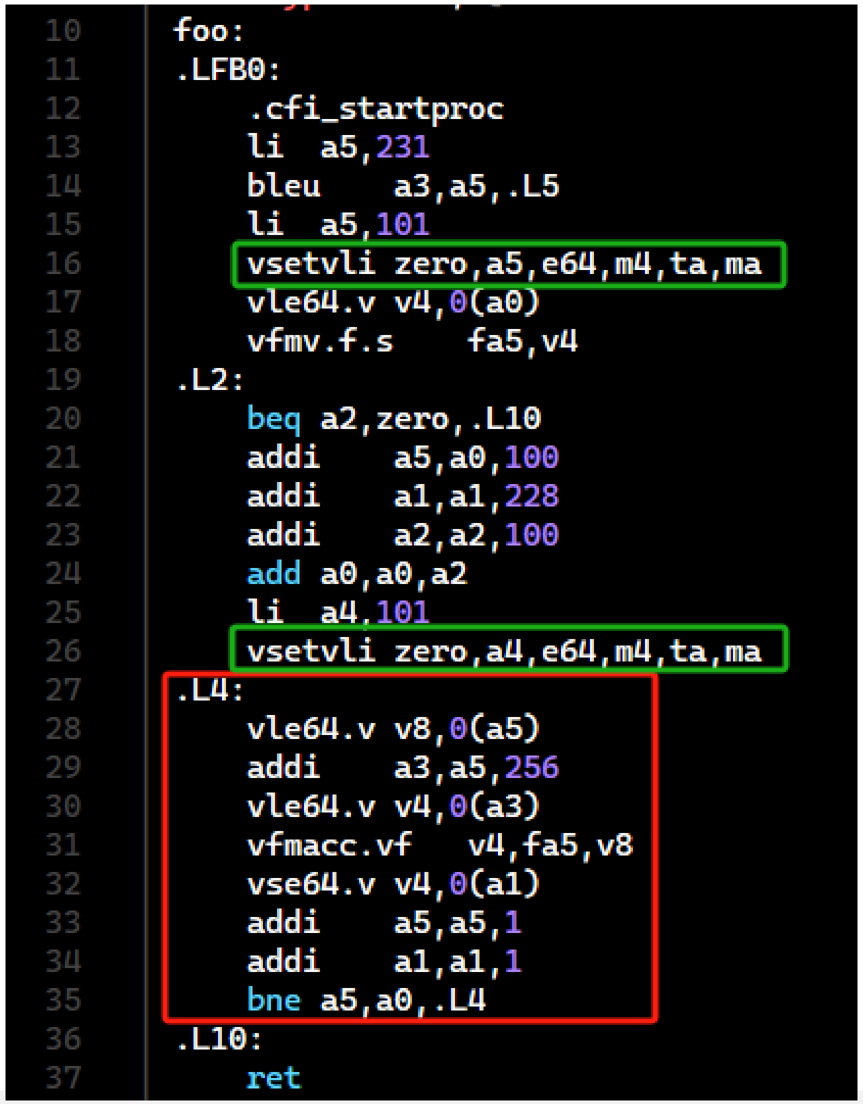
```c
void
vec_shift (int64_t *out, int64_t *a,
                uint32_t size)
{
  for (uint32_t  i = 0; i < size; i++)
    out[i] = a[i] << 7;
}
```



```asm
vec_shift:
.LFB0:
	.cfi_startproc
	beq a2,zero,.L12
	addiw	a5,a2,-1
	li	a4,4
	bleu	a5,a4,.L3
	addi	a5,a1,8
	beq a0,a5,.L3
	slli	a2,a2,32
	srli	a2,a2,32
.L4:
	vsetvli a5,a2,e64,m1,ta,ma
	slli	a4,a5,3
	vle64.v v1,0(a1)
	vsll.vi v1,v1,7
	vse64.v v1,0(a0)
	add a1,a1,a4
	add a0,a0,a4
	sub a2,a2,a5
	bne a2,zero,.L4
	ret
```

Advanced VSETVL (default on) - by RiVAI

```c
void foo (int8_t * restrict in, int8_t * restrict out,
          int n, int cond)
{
  size_t vl = 101; double f = -0.94;
  vfloat64m4_t v2, v3, v4;
  if (cond > 231) {
      v2 = __riscv_vle64_v_f64m4 (in, vl);
      f = __riscv_vfmv_f_s_f64m4_f64 (v2);
  }
  for (size_t i = 0; i < n; i++) {
      v3 = __riscv_vle64_v_f64m4 (in + i + 32, vl);
      v4 = __riscv_vle64_v_f64m4 (in + i + 64, vl);
      v4 = __riscv_vfmacc_vf_f64m4 (v4, f, v3, vl);
      __riscv_vse64_v_f64m4 (out + 128, v4, vl);
  }
}
```



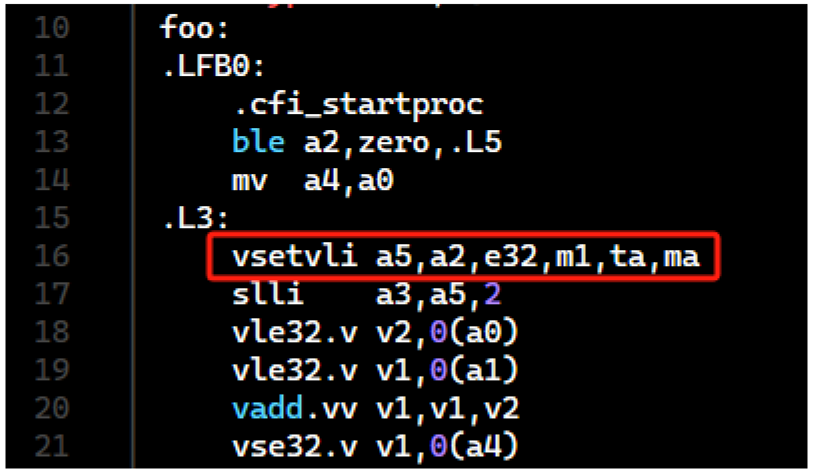
```asm
foo:
.LFB0:
	.cfi_startproc
	li	a5,231
	bleu	a3,a5,.L5
	li	a5,101
	vsetvli	zero,a5,e64,m4,ta,ma
	vle64.v	v4,0(a0)
	vfmv.f.s	fa5,v4
.L2:
	beq	a2,zero,.L10
	addi	a5,a0,100
	addi	a1,a1,228
	addi	a2,a2,100
	add	a0,a0,a2
	li	a4,101
	vsetvli	zero,a4,e64,m4,ta,ma
.L4:
	vle64.v	v8,0(a5)
	addi	a3,a5,256
	vle64.v	v4,0(a3)
	vfmacc.vf	v4,fa5,v8
	vse64.v	v4,0(a1)
	addi	a5,a5,1
	addi	a1,a1,1
	bne	a5,a0,.L4
.L10:
	ret
```

Dynamic LMUL - by RiVAI

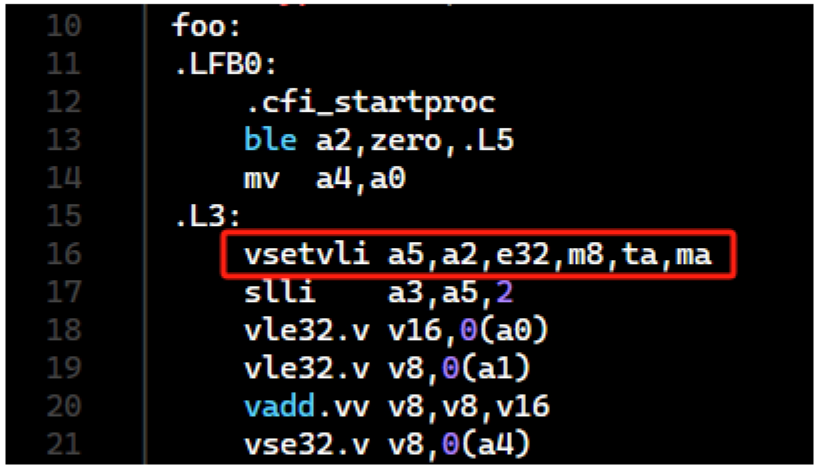
-mrvv-max-lmul=**m1 (default)**

-mrvv-max-lmul=**dynamic**

```
1.void
2.foo (int32_t *__restrict a,
3.       int32_t *__restrict b, int n)
4.{
5.  for (int i = 0; i < n; i++)
6.    a[i] = a[i] + b[i];
7.}
```



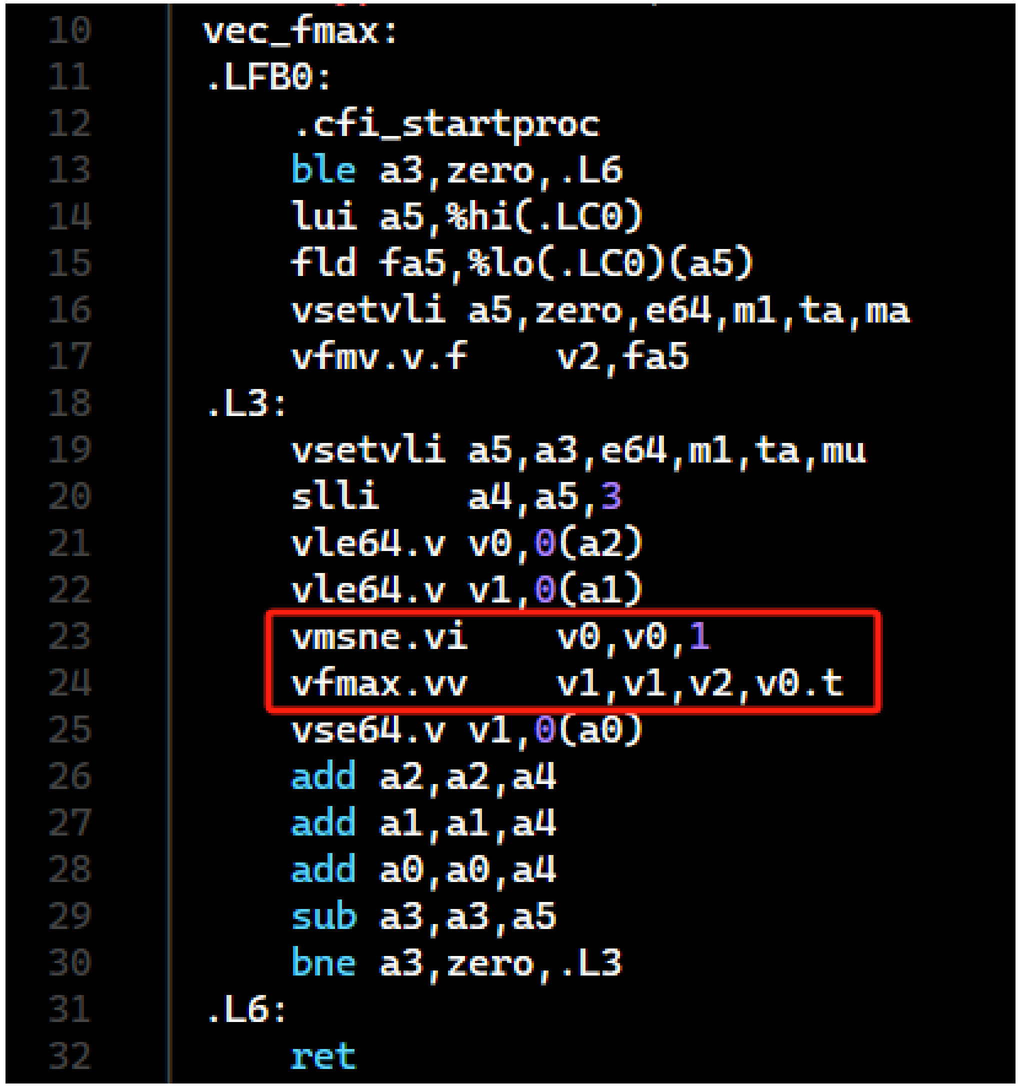
```
foo:
.LFB0:
	.cfi_startproc
	ble a2,zero,.L5
	mv  a4,a0
.L3:
	vsetvli a5,a2,e32,m1,ta,ma
	slli    a3,a5,2
	vle32.v v2,0(a0)
	vle32.v v1,0(a1)
	vadd.vv v1,v1,v2
	vse32.v v1,0(a4)
```



```
foo:
.LFB0:
	.cfi_startproc
	ble a2,zero,.L5
	mv  a4,a0
.L3:
	vsetvli a5,a2,e32,m8,ta,ma
	slli    a3,a5,2
	vle32.v v16,0(a0)
	vle32.v v8,0(a1)
	vadd.vv v8,v8,v16
	vse32.v v8,0(a4)
```

Conditional (default on) - by RiVAI and Ventana Micro

```c
void
vec_fmax (double *__restrict a,
          double *__restrict b,
          int64_t *pred, int n)
{
  for (int i = 0; i < n; i++)
    {
      if (pred[i] != 1)
        a[i] = __builtin_fmaxf64 (b[i], 3.0);
       else
        a[i] = b[i];
    }
}
```



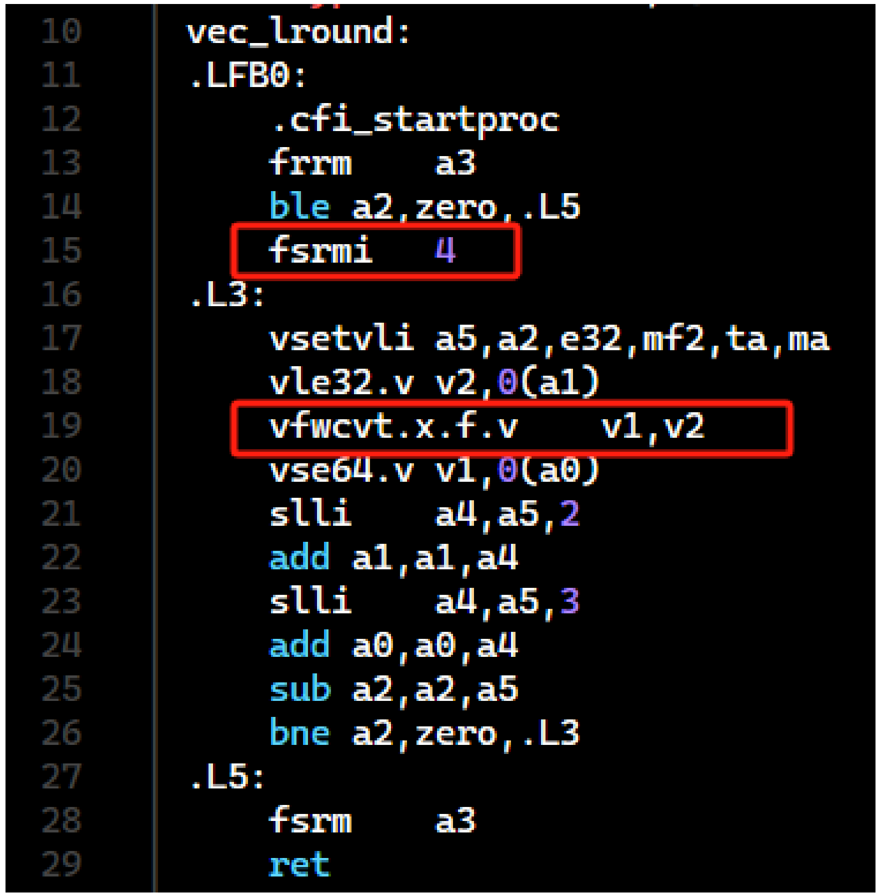
```
vec_fmax:
.LFB0:
	.cfi_startproc
	ble a3,zero,.L6
	lui a5,%hi(.LC0)
	fld fa5,%lo(.LC0)(a5)
	vsetvli a5,zero,e64,m1,ta,ma
	vfmv.v.f	v2,fa5
.L3:
	vsetvli a5,a3,e64,m1,ta,mu
	slli	a4,a5,3
	vle64.v v0,0(a2)
	vle64.v v1,0(a1)
	vmsne.vi	v0,v0,1
	vfmax.vv	v1,v1,v2,v0.t
	vse64.v v1,0(a0)
	add a2,a2,a4
	add a1,a1,a4
	add a0,a0,a4
	sub a3,a3,a5
	bne a3,zero,.L3
.L6:
	ret
```

Rounding - by Intel

-ffast-math

round/ceil/floor/rint/trunc/nearbyint

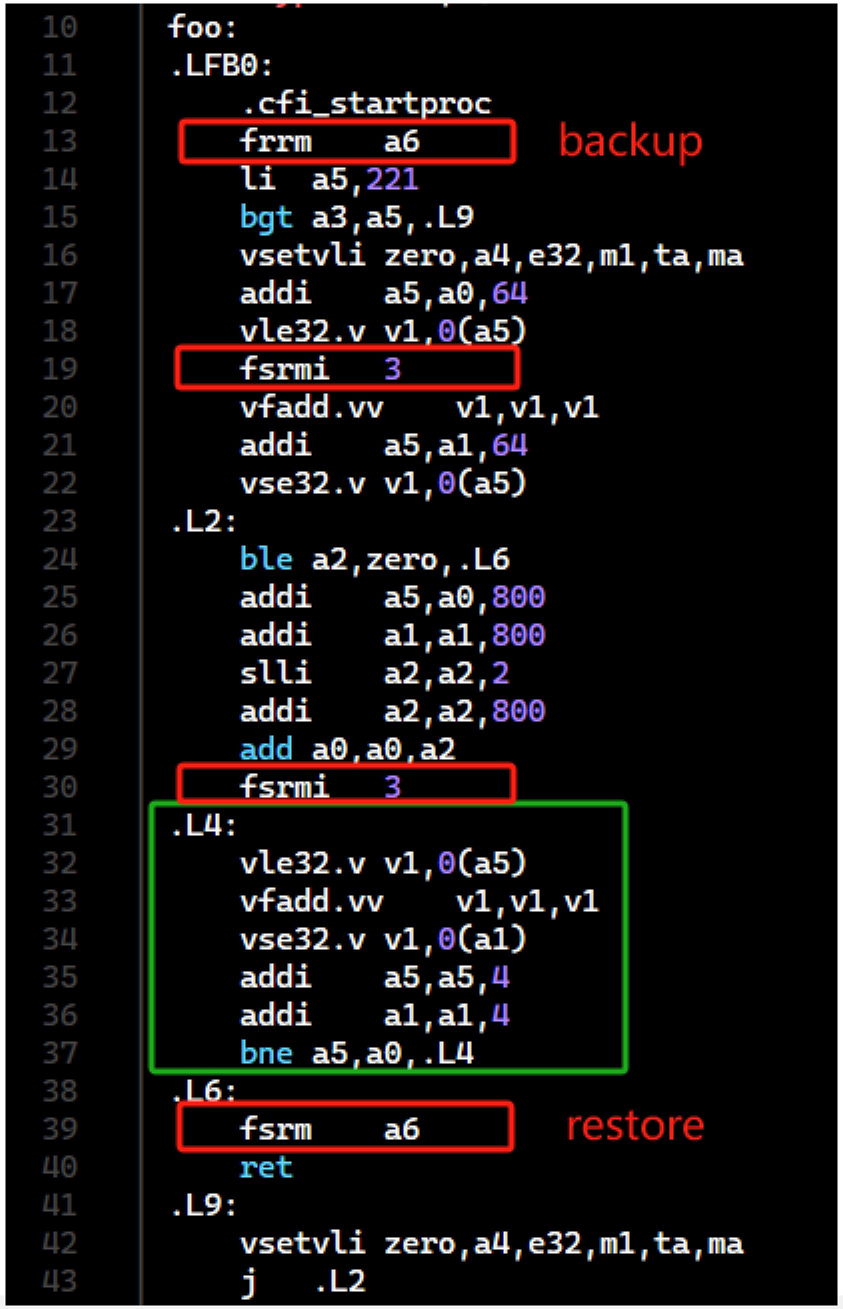
```
1.void
2.vec_lround (long *out, float *a, int n)
3.{
4.  for (int i = 0; i < n; i++)
5.     out[i] = __builtin_lroundf (a[i]);
6.}
```



```
vec_lround:
.LFB0:
        .cfi_startproc
        frrm     a3
        ble a2,zero,.L5
        fsrmi    4
.L3:
        vsetvli a5,a2,e32,mf2,ta,ma
        vle32.v v2,0(a1)
        vfwcvt.x.f.v     v1,v2
        vse64.v v1,0(a0)
        slli     a4,a5,2
        add a1,a1,a4
        slli     a4,a5,3
        add a0,a0,a4
        sub a2,a2,a5
        bne a2,zero,.L3
.L5:
        fsrm     a3
        ret
```

Advanced FRM (default on) - by Intel

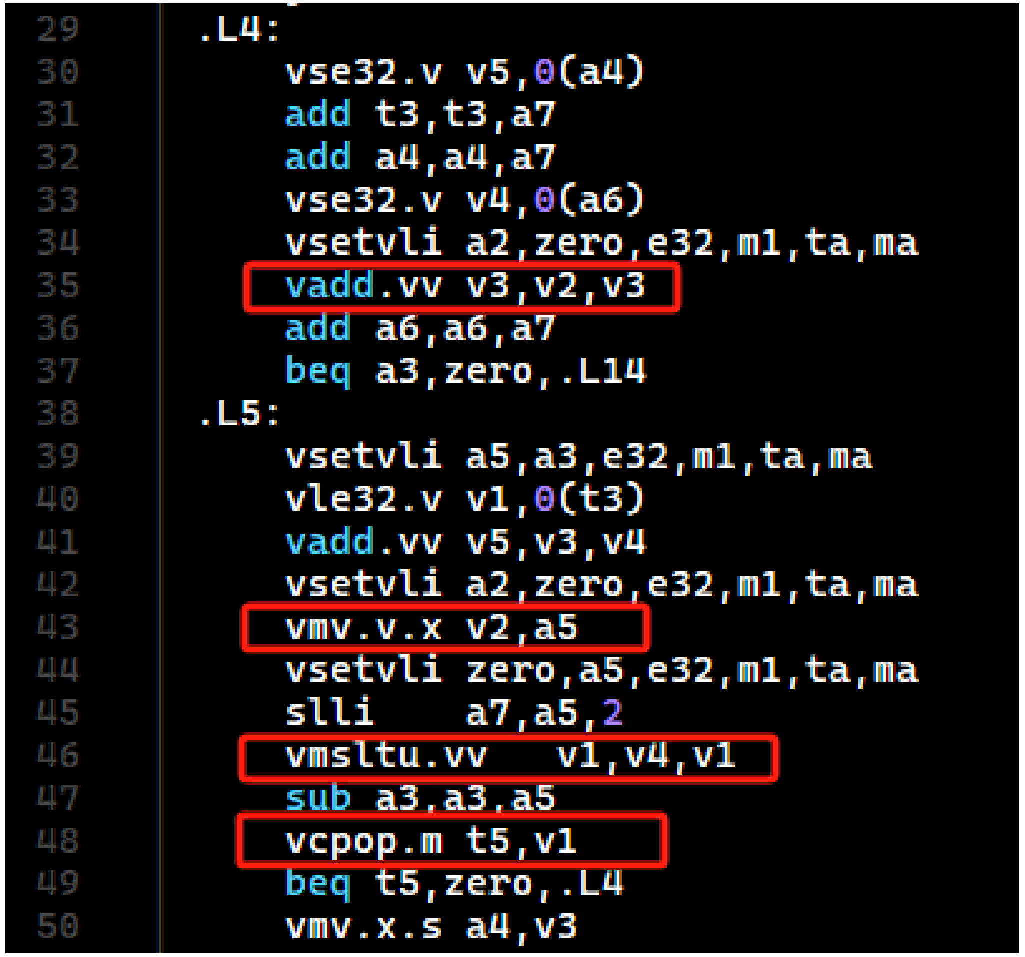
```c
void
foo (float *in, float *out, int n, int cond, size_t vl)
{
  vfloat32m1_t v, result;
  if (cond < 0xde)
    {
      v = __riscv_vle32_v_f32m1 (in + 16, vl);
      result = __riscv_vfadd_vv_f32m1_rm (v, v, 3, vl);
      __riscv_vse32_v_f32m1 (out + 16, result, vl);
    }
  for (int i = 0; i < n; i++)
    {
      v = __riscv_vle32_v_f32m1 (in + i + 200, vl);
      result = __riscv_vfadd_vv_f32m1_rm (v, v, 3, vl);
      __riscv_vse32_v_f32m1 (out + i + 200, result, vl);
    }
}
```



```asm
foo:
.LFB0:
	.cfi_startproc
	frrm	a6
	li	a5,221
	bgt	a3,a5,.L9
	vsetvli	zero,a4,e32,m1,ta,ma
	addi	a5,a0,64
	vle32.v	v1,0(a5)
	fsrmi	3
	vfadd.vv	v1,v1,v1
	addi	a5,a1,64
	vse32.v	v1,0(a5)
.L2:
	ble	a2,zero,.L6
	addi	a5,a0,800
	addi	a1,a1,800
	slli	a2,a2,2
	addi	a2,a2,800
	add	a0,a0,a2
	fsrmi	3
.L4:
	vle32.v	v1,0(a5)
	vfadd.vv	v1,v1,v1
	vse32.v	v1,0(a1)
	addi	a5,a5,4
	addi	a1,a1,4
	bne	a5,a0,.L4
.L6:
	fsrm	a6
	ret
.L9:
	vsetvli	zero,a4,e32,m1,ta,ma
	j	.L2
```

Early Exit (GCC-15) – by RiVAI and Intel

```c
#define N 901
unsigned vect_a[N];
unsigned vect_b[N];
void test (unsigned x, int n)
{
  for (int i = 0; i < n; i++)
  {
    vect_b[i] = x + i;
    if (vect_a[i] > x)
      break;
    vect_a[i] = x;
  }
}
```

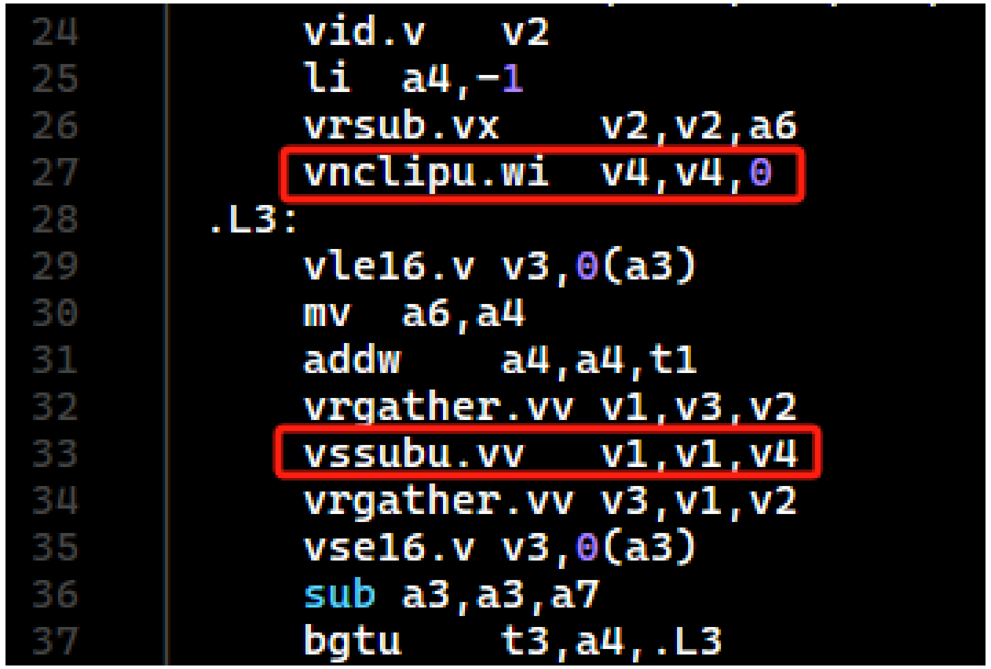


```
.L4:
        vse32.v v5,0(a4)
        add t3,t3,a7
        add a4,a4,a7
        vse32.v v4,0(a6)
        vsetvli a2,zero,e32,m1,ta,ma
        vadd.vv v3,v2,v3
        add a6,a6,a7
        beq a3,zero,.L14
.L5:
        vsetvli a5,a3,e32,m1,ta,ma
        vle32.v v1,0(t3)
        vadd.vv v5,v3,v4
        vsetvli a2,zero,e32,m1,ta,ma
        vmv.v.x v2,a5
        vsetvli zero,a5,e32,m1,ta,ma
        slli    a7,a5,2
        vmsltu.vv    v1,v4,v1
        sub a3,a3,a5
        vcpop.m t5,v1
        beq t5,zero,.L4
        vmv.x.s a4,v3
```

Saturation ALU (GCC-15) - by Intel

Sub

```
void
zip (uint16_t *x, uint32_t b, unsigned n)
{
  uint32_t a;
  uint16_t *p = x;
  do {
    a = *--p;
    *p = (uint16_t)(a >= b ? a - b : 0);
  } while (--n);
}
```

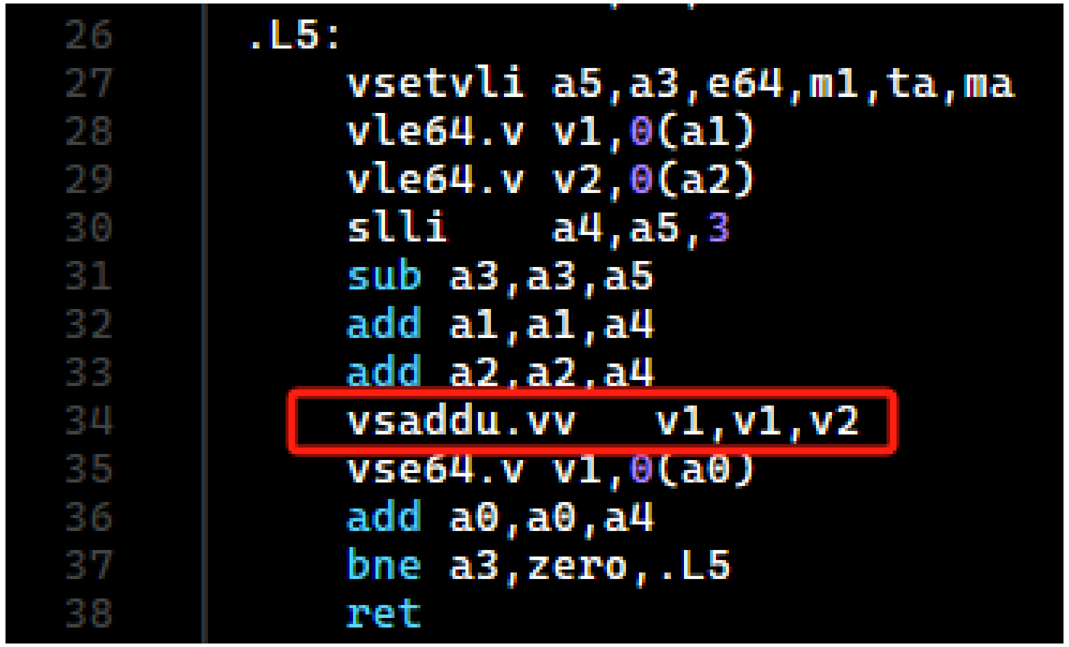


```
        vid.v   v2
        li   a4,-1
        vrsub.vx    v2,v2,a6
        vnclipu.wi  v4,v4,0
.L3:
        vle16.v v3,0(a3)
        mv   a6,a4
        addw    a4,a4,t1
        vrgather.vv v1,v3,v2
        vssubu.vv   v1,v1,v4
        vrgather.vv v3,v1,v2
        vse16.v v3,0(a3)
        sub a3,a3,a7
        bgtu    t3,a4,.L3
```

Saturation ALU (GCC-15) - by Intel

Add

```c
void
vec_sat_add (uint64_t *out, uint64_t *a,
             uint64_t *b, uint32_t size)
{
  for (uint32_t i = 0; i < size; i++)
    {
      if ((a[i] + b[i]) >= a[i])
        out[i] = a[i] + b[i];
      else
        out[i] = -1;
    }
}
```

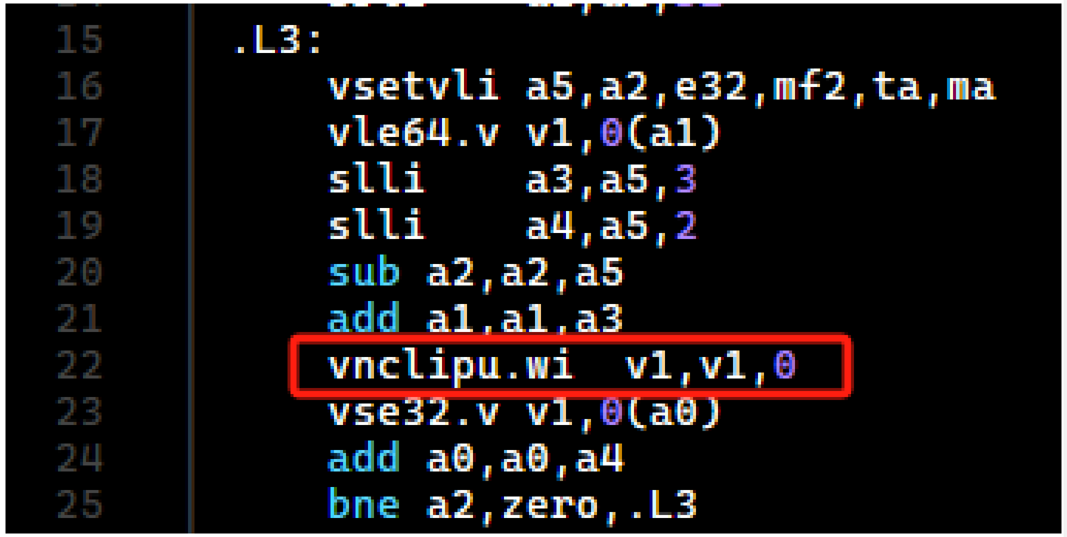


```asm
.L5:
    vsetvli a5,a3,e64,m1,ta,ma
    vle64.v v1,0(a1)
    vle64.v v2,0(a2)
    slli    a4,a5,3
    sub a3,a3,a5
    add a1,a1,a4
    add a2,a2,a4
    vsaddu.vv   v1,v1,v2
    vse64.v v1,0(a0)
    add a0,a0,a4
    bne a3,zero,.L5
    ret
```

Saturation ALU (GCC-15) - by Intel

Truncate

```
void
vec_sat_trunc (uint32_t *out, uint64_t *a,
               uint32_t size)
{
  for (uint32_t i = 0; i < size; i++)
    {
      uint64_t x = a[i];
      bool overflow = x > (uint64_t)(uint32_t)(-1);
      if (overflow)
        out[i] = -1;
      else
        out[i] = (uint32_t)x;
    }
}
```



```
.L3:
        vsetvli a5,a2,e32,mf2,ta,ma
        vle64.v v1,0(a1)
        slli    a3,a5,3
        slli    a4,a5,2
        sub a2,a2,a5
        add a1,a1,a3
        vnclipu.wi  v1,v1,0
        vse32.v v1,0(a0)
        add a0,a0,a4
        bne a2,zero,.L3
```

More in GCC-15

- Stride Load/Store.
- Vector Register Overlap.
- VLS Calling Convention.

RISE
RISC-V Software Ecosystem

- https://riseproject.dev

RISE is focused on positive and transparent collaborations with upstream projects to deliver commercial-ready software for various use cases

How: Align on highest priorities & avoid (accidental) duplication of work

Goal: Accelerate open source SW for RISC-V architecture

https://www.intel.com/content/www/us/en/developer/articles/community/rising-to-the-challenge-risc-v-software-readiness.html

Finding more interesting topics from Intel on RISC-V summit China 2024

Topic	When & Where
UXL 软件栈和 RISC-V 的初步探索	August 22 16:45 主会场A
LLVM 工具链 RISC-V 构建实现及其性能优化现状分析与未来展望	August 23 9:40 主会场A
GCC RVV 自动向量化及其应用	August 23 10:00 主会场A
Enhancing RISC-V Security with SBI Secure Service APIs	August 23 10:40 主会场B
Enabling Hardware Sampling Based PGO for RISC-V Platform	August 23 11:40 主会场A
利用 WASM 技术解决多种 ISA 的挑战	August 23 14:20 主会场B
HVP: Hardware Accelerated RISC-V Android Emulator	August 23 14:50 主会场A
Leverage BRS standard to improve RISC-V SW compatibility	August 23 17:30 主会场A
Soft-ISA: kernel built-in emulation engine to extend RISC-V silicon ISA capability	August 23 17:40 主会场A

intel®