OpenJDK Vector API在RISC-V上的实现

RuyiSDK OpenJDK Team

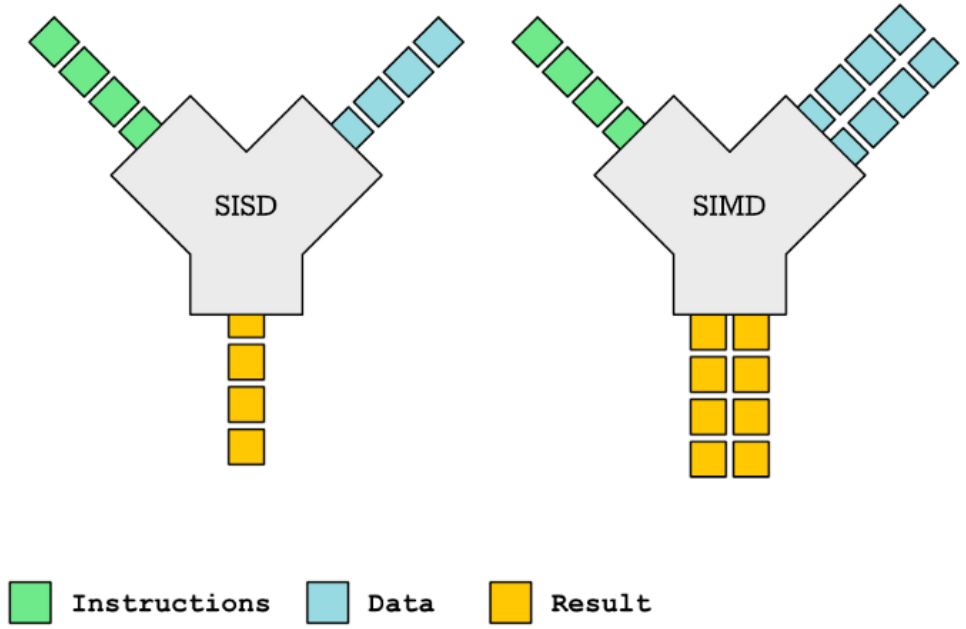
张定立 dingli@iscas.ac.cn

曹贵 caogui@iscas.ac.cn

Table of Contents

OpenJDK Vector API 介绍

SISD & SIMD



参考资料：

[1] https://ichi.pro/de/apples-m1-secret-coprozessor-111709032268257

OpenJDK Vector API 介绍

Vector API JEPs

Project Panama

- **JEP 338: Vector API (Incubator) – JDK16**
- **JEP 414: Vector API (Second Incubator**
- **JEP 417: Vector API (Third Incubator)**
- **JEP 426: Vector API (Fourth Incubator)**
- **JEP 438: Vector API (Fifth Incubator)**
- **JEP 448: Vector API (Sixth Incubator) – JDK21**

OpenJDK Vector API 介绍

Vector API 示例

```java
public static int[] simpleSum(int[] a, int[] b) {
    var c = new int[a.length];
    for (var i = 0; i < a.length; i++) {
        c[i] = a[i] + b[i];
    }
    return c;
}
```

原始代码

```java
private static final VectorSpecies<Integer> SPECIES = IntVector.SPECIES_PREFERRED;
public static int[] vectorSum(int[] a, int[] b) {
    var c = new int[a.length];
    var upperBound = SPECIES.loopBound(a.length);

    var i = 0;
    for (; i < upperBound; i += SPECIES.length()) {
        var va = IntVector.fromArray(SPECIES, a, i);
        var vb = IntVector.fromArray(SPECIES, b, i);
        var vc = va.add(vb);
        vc.intoArray(c, i);
    }
    // Compute elements not fitting in the vector alignment.
    for (; i < a.length; i++) {
        c[i] = a[i] + b[i];
    }

    return c;

}
```

通过 Vector API 转换为数据并行加速代码

参考资料：

[1] https://medium.com/@Styp/java-18-vector-api-do-we-get-free-speed-up-c4510eda50d2

OpenJDK Vector API 介绍

RVV简介

- RVV（RISC-V “V” Extension）是RISC-V的标准扩展。
- 主要由 Krste Asanovic 设计。
- RVV目前为 1.0 版本，frozen for public review
- https://github.com/riscv/riscv-v-spec
- 目前OpenJDK只支持RVV1.0[1]

参考资料：

[1] https://github.com/openjdk/jdk/pull/10878#issuecomment-1294548949

OpenJDK Vector API 介绍

开发环境及测试状态

- **Emulation：QEMU 7.0+**
- **Compiler：GCC**
- **JTReg tier1-tier3 w/ and w/o UseRVV**
- **75 tests under incubator/vector w/ and w/o UseRVV**

OpenJDK Vector API 介绍

Vector API以外RVV的使用

```
void C2_MacroAssembler::arrays_equals_v(Register a1, Register a2, Register result,
                                        Register cnt1, int elem_size) {
  Label DONE;
  Register tmp1 = t0;
  Register tmp2 = t1;
  Register cnt2 = tmp2;
  int length_offset = arrayOopDesc::length_offset_in_bytes();
  int base_offset = arrayOopDesc::base_offset_in_bytes(elem_size == 2 ? T_CHAR : T_BYTE);

  BLOCK_COMMENT("arrays_equals_v {");

  // if (a1 == a2), return true
  mv(result, true);
  beq(a1, a2, DONE);

  mv(result, false);
  // if a1 == null or a2 == null, return false
  beqz(a1, DONE);
  beqz(a2, DONE);
  // if (a1.length != a2.length), return false
  lwu(cnt1, Address(a1, length_offset));
  lwu(cnt2, Address(a2, length_offset));
  bne(cnt1, cnt2, DONE);

  la(a1, Address(a1, base_offset));
  la(a2, Address(a2, base_offset));

  element_compare(a1, a2, result, cnt1, tmp1, tmp2, v2, v4, v2, elem_size == 1, DONE);

  bind(DONE);

  BLOCK_COMMENT("} arrays_equals_v");
}
```

Array equals

```
// used by C2 ClearArray patterns.
// base: Address of a buffer to be zeroed
// cnt: Count in HeapWords
//
// base, cnt, v4, v5, v6, v7 and t0 are clobbered.
void C2_MacroAssembler::clear_array_v(Register base, Register cnt) {
  Label loop;

  // making zero words
  vsetvli(t0, cnt, Assembler::e64, Assembler::m4);
  vxor_vv(v4, v4, v4);

  bind(loop);
  vsetvli(t0, cnt, Assembler::e64, Assembler::m4);
  vse64_v(v4, base);
  sub(cnt, cnt, t0);
  shadd(base, t0, base, t0, 3);
  bnez(cnt, loop);
}
```

Clear Array

OpenJDK Vector API 介绍

封装的高频调用RISC-V指令

封装的vset指令

```
// Set vl and vtype for full and partial vector operations.
// (vma = mu, vta = tu, vill = false)
void C2_MacroAssembler::vsetvli_helper(BasicType bt, int vector_length, LMUL vlmul, Register tmp) {
  Assembler::SEW sew = Assembler::elemtype_to_sew(bt);
  if (vector_length <= 31) {
    vsetivli(tmp, vector_length, sew, vlmul);
  } else if (vector_length == (MaxVectorSize / type2aelembytes(bt))) {
    vsetvli(tmp, x0, sew, vlmul);
  } else {
    mv(tmp, vector_length);
    vsetvli(tmp, tmp, sew, vlmul);
  }
}
```

封装的vector store指令

```
void vsex_v(VectorRegister store_data, Register base, Assembler::SEW sew, VectorMask vm = unmasked) {
  switch (sew) {
    case Assembler::e64:
      vse64_v(store_data, base, vm);
      break;
    case Assembler::e32:
      vse32_v(store_data, base, vm);
      break;
    case Assembler::e16:
      vse16_v(store_data, base, vm);
      break;
    case Assembler::e8: // fall through
    default:
      vse8_v(store_data, base, vm);
      break;
  }
}
```

OpenJDK Vector API 介绍

Vector API Opcode类型

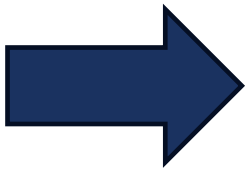
- **Arithmetic**
- **Compress/Expand**
- **Reduction**
- **Shift**
- **Load/Store**
- **Rearrange/Shuffle**
- **Cast**
- **Mask**

```
bool MatchRule::is_vector() const {
  static const char *vector_list[] = {
    "AddVB","AddVS","AddVI","AddVL","AddVF","AddVD",
    "SubVB","SubVS","SubVI","SubVL","SubVF","SubVD",
    "MulVB","MulVS","MulVI","MulVL","MulVF","MulVD",
    "DivVF","DivVD",
    "AbsVB","AbsVS","AbsVI","AbsVL","AbsVF","AbsVD",
    "NegVF","NegVD","NegVI","NegVL",
    "SqrtVD","SqrtVF",
    "AndV" ,"XorV" ,"OrV",
    "MaxV", "MinV",
    "CompressV", "ExpandV", "CompressM", "CompressBitsV", "ExpandBitsV",
    "AddReductionVI", "AddReductionVL",
    "AddReductionVF", "AddReductionVD",
    "MulReductionVI", "MulReductionVL",
    "MulReductionVF", "MulReductionVD",
    "MaxReductionV", "MinReductionV",
    "AndReductionV", "OrReductionV", "XorReductionV",
    "MulAddVS2VI", "MacroLogicV",
    "LShiftCntV","RShiftCntV",
    "LShiftVB","LShiftVS","LShiftVI","LShiftVL",
    "RShiftVB","RShiftVS","RShiftVI","RShiftVL",
    "URShiftVB","URShiftVS","URShiftVI","URShiftVL",
    "ReplicateB","ReplicateS","ReplicateI","ReplicateL","ReplicateF","ReplicateD","ReverseV","ReverseBytesV",
    "RoundDoubleModeV","RotateLeftV" , "RotateRightV", "LoadVector","StoreVector",
    "LoadVectorGather", "StoreVectorScatter", "LoadVectorGatherMasked", "StoreVectorScatterMasked",
    "VectorTest", "VectorLoadMask", "VectorStoreMask", "VectorBlend", "VectorInsert",
    "VectorRearrange","VectorLoadShuffle", "VectorLoadConst",
    "VectorCastB2X", "VectorCastS2X", "VectorCastI2X",
    "VectorCastL2X", "VectorCastF2X", "VectorCastD2X", "VectorCastF2HF", "VectorCastHF2F",
    "VectorUCastB2X", "VectorUCastS2X", "VectorUCastI2X",
    "VectorMaskWrapper","VectorMaskCmp","VectorReinterpret","LoadVectorMasked","StoreVectorMasked",
    "FmaVD","FmaVF","PopCountVI","PopCountVL","PopulateIndex","VectorLongToMask",
    "CountLeadingZerosV", "CountTrailingZerosV", "SignumVF", "SignumVD",
    // Next are vector mask ops.
    "MaskAll", "AndVMask", "OrVMask", "XorVMask", "VectorMaskCast",
    "RoundVF", "RoundVD",
    // Next are not supported currently.
    "PackB","PackS","PackI","PackL","PackF","PackD","Pack2L","Pack2D",
    "ExtractB","ExtractUB","ExtractC","ExtractS","ExtractI","ExtractL","ExtractF","ExtractD"
  };
```

OpenJDK Vector API 介绍

RISC-V Port Supported vector match rule

```
// Vector API specific
case Op_AndReductionV:
case Op_OrReductionV:
case Op_XorReductionV:
case Op_LoadVectorGather:
case Op_StoreVectorScatter:
case Op_VectorBlend:
case Op_VectorCast:
case Op_VectorCastB2X:
case Op_VectorCastD2X:
case Op_VectorCastF2X:
case Op_VectorCastI2X:
case Op_VectorCastL2X:
case Op_VectorCastS2X:
case Op_VectorInsert:
case Op_VectorLoadConst:
case Op_VectorLoadMask:
case Op_VectorLoadShuffle:
case Op_VectorMaskCmp:
case Op_VectorRearrange:
case Op_VectorReinterpret:
case Op_VectorStoreMask:
case Op_VectorTest:
  return false;
```



```
bool Matcher::match_rule_supported_vector(int opcode, int vlen, BasicType bt) {
  if (!UseRVV) {
    return false;
  }

  if (!match_rule_supported(opcode) || !vector_size_supported(bt, vlen)) {
    return false;
  }

  switch (opcode) {
    case Op_VectorMaskLastTrue:
      if (!UseZbb || vlen > XLEN) {
        return false;
      }
      break;
    case Op_VectorMaskToLong:
    case Op_VectorLongToMask:
      if (vlen > XLEN) {
        return false;
      }
      break;
    default:
      break;
  }
  return true;
}
```

8276799: Implementation of JEP 422: Linux/RISC-V Port

8307609: RISC-V: Added support for Extract, Compress, Expand and other nodes for Vector API

OpenJDK Vector API 实现示例

Int类型的AddReductionVI实现

```
instruct reduce_addI(iRegINoSp dst, iRegIorL2I src1, vReg src2, vReg tmp) %{
  predicate(Matcher::vector_element_basic_type(n->in(2)) == T_BYTE ||
            Matcher::vector_element_basic_type(n->in(2)) == T_SHORT ||
            Matcher::vector_element_basic_type(n->in(2)) == T_INT);
  match(Set dst (AddReductionVI src1 src2));
  effect(TEMP tmp);
  ins_cost(VEC_COST);
  format %{ "reduce_addI $dst, $src1, $src2\t# KILL $tmp" %}
  ins_encode %{
    BasicType bt = Matcher::vector_element_basic_type(this, $src2);
    __ reduce_integral_v($dst$$Register, $src1$$Register,
                         as_VectorRegister($src2$$reg), as_VectorRegister($tmp$$reg),
                         this->ideal_Opcode(), bt, Matcher::vector_length(this, $src2));
  %}
  ins_pipe(pipe_slow);
%}
```

AddReductionVI的C2 instruct

```
void C2_MacroAssembler::reduce_integral_v(Register dst, Register src1,
                                          VectorRegister src2, VectorRegister tmp,
                                          int opc, BasicType bt, int vector_length, VectorMask vm) {
  assert(bt == T_BYTE || bt == T_SHORT || bt == T_INT || bt == T_LONG, "unsupported element type");
  vsetvli_helper(bt, vector_length);
  vmv_s_x(tmp, src1);
  switch (opc) {
    case Op_AddReductionVI:
    case Op_AddReductionVL:
      vredsum_vs(tmp, src2, tmp, vm);
      break;
    case Op_AndReductionV:
      vredand_vs(tmp, src2, tmp, vm);
      break;
    case Op_OrReductionV:
      vredor_vs(tmp, src2, tmp, vm);
      break;
    case Op_XorReductionV:
      vredxor_vs(tmp, src2, tmp, vm);
      break;
    case Op_MaxReductionV:
      vredmax_vs(tmp, src2, tmp, vm);
      break;
    case Op_MinReductionV:
      vredmin_vs(tmp, src2, tmp, vm);
      break;
    default:
      ShouldNotReachHere();
  }
  vmv_x_s(dst, tmp);
}
```

封装的reduce函数

OpenJDK Vector API 实现示例

Vector API IntVector提供的reduceLanes 接口

```
 * @param op the operation used to combine lane values
 * @return the accumulated result
 * @throws UnsupportedOperationException if this vector does
 *         not support the requested operation
 * @see #reduceLanes(VectorOperators.Associative,VectorMask)
 * @see #add(Vector)
 * @see #mul(Vector)
 * @see #min(Vector)
 * @see #max(Vector)
 * @see #and(Vector)
 * @see #or(Vector)
 * @see VectorOperators#XOR
 * @see VectorOperators#FIRST_NONZERO
 */
public abstract int reduceLanes(VectorOperators.Associative op);
```

OpenJDK Vector API 实现示例

C2 未提供Vector API实现的调用

```
@IntrinsicCandidate
public static
<V extends Vector<E>,
 M extends VectorMask<E>,
 E>
long reductionCoerced(int oprId,
                      Class<? extends V> vClass, Class<? extends M> mClass, Class<E> eClass,
                      int length,
                      V v, M m,
                      ReductionOperation<V, M> defaultImpl) {
    assert isNonCapturingLambda(defaultImpl) : defaultImpl;
    return defaultImpl.apply(v, m);
}
```

Reduction的方法签名

```
private static ReductionOperation<IntVector, VectorMask<Integer>> reductionOperations(int opc_) {
    switch (opc_) {
        case VECTOR_OP_ADD: return (v, m) ->
                toBits(v.rOp((int)0, m, (i, a, b) -> (int)(a + b)));
        case VECTOR_OP_MUL: return (v, m) ->
                toBits(v.rOp((int)1, m, (i, a, b) -> (int)(a * b)));
        case VECTOR_OP_MIN: return (v, m) ->
                toBits(v.rOp(MAX_OR_INF, m, (i, a, b) -> (int) Math.min(a, b)));
        case VECTOR_OP_MAX: return (v, m) ->
                toBits(v.rOp(MIN_OR_INF, m, (i, a, b) -> (int) Math.max(a, b)));
        case VECTOR_OP_AND: return (v, m) ->
                toBits(v.rOp((int)-1, m, (i, a, b) -> (int)(a & b)));
        case VECTOR_OP_OR: return (v, m) ->
                toBits(v.rOp((int)0, m, (i, a, b) -> (int)(a | b)));
        case VECTOR_OP_XOR: return (v, m) ->
                toBits(v.rOp((int)0, m, (i, a, b) -> (int)(a ^ b)));
        default: return null;
    }
}
```

Reduction的默认实现

OpenJDK Vector API 实现示例

C2 提供Vector API 实现的调用

```
// public static
// <V extends Vector<E>,
//  M extends VectorMask<E>,
//  E>
// long reductionCoerced(int oprId, Class<? extends V> vectorClass, Class<? extends M> maskClass,
//                       Class<E> elementType, int length, V v, M m,
//                       ReductionOperation<V, M> defaultImpl)
bool LibraryCallKit::inline_vector_reduction() {
  const TypeInt*     opr          = gvn().type(argument(0))->isa_int();
  const TypeInstPtr* vector_klass = gvn().type(argument(1))->isa_instptr();
  const TypeInstPtr* mask_klass   = gvn().type(argument(2))->isa_instptr();
  const TypeInstPtr* elem_klass   = gvn().type(argument(3))->isa_instptr();
  const TypeInt*     vlen         = gvn().type(argument(4))->isa_int();

  if (opr == nullptr || vector_klass == nullptr || elem_klass == nullptr || vlen == nullptr ||
      !opr->is_con() || vector_klass->const_oop() == nullptr || elem_klass->const_oop() == nullptr || !vlen->is_con()) {
    if (C->print_intrinsics()) {
      tty->print_cr("  ** missing constant: opr=%s vclass=%s etype=%s vlen=%s",
                    NodeClassNames[argument(0)->Opcode()],
                    NodeClassNames[argument(1)->Opcode()],
                    NodeClassNames[argument(3)->Opcode()],
                    NodeClassNames[argument(4)->Opcode()]);
    }
    return false; // not enough info for intrinsification
  }
```

Reduction的Vector API实现

OpenJDK Vector API 实现示例

RISC-V Port Supported vector masked match rule

```
const bool Matcher::match_rule_supported_vector_masked(int opcode, int vlen, BasicType bt) {
  return false;
}
```

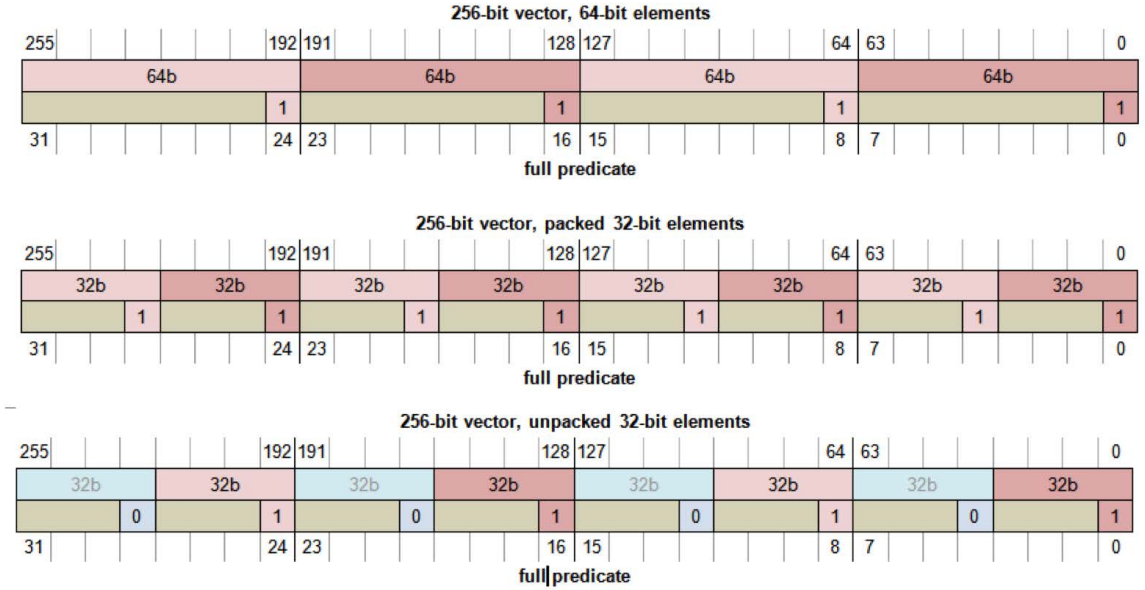
8276799: Implementation of JEP 422: Linux/RISC-V Port

```
bool Matcher::match_rule_supported_vector_masked(int opcode, int vlen, BasicType bt) {
  if (!UseRVV) {
    return false;
  }
  return match_rule_supported_vector(opcode, vlen, bt);
}
```

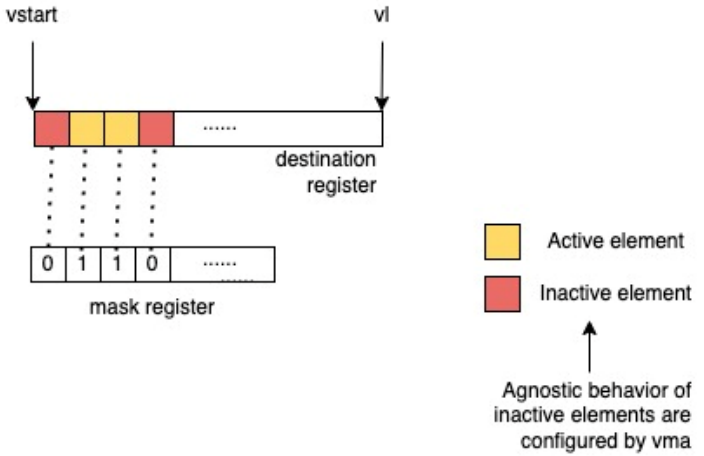
8307609: RISC-V: Added support for Extract, Compress, Expand and other nodes for Vector API

OpenJDK Vector API 实现示例

SVE与RVV的mask操作对比



SVE predicate register



RVV mask register

参考资料：

[1] https://arxiv.org/pdf/1803.06185.pdf

[2] https://eopxd.com/2022/07/01/rvv-policy/

OpenJDK Vector API 实现示例

SVE与RVV的mask操作对比(VectorLongToMask)

```
// Unpack the mask, a long value in src, into predicate register dst based on the
// corresponding data type. Note that dst can support at most 64 lanes.
// Below example gives the expected dst predicate register in different types, with
// a valid src(0x658D) on a 1024-bit vector size machine.
// BYTE:  dst = 0x00 00 00 00 00 00 00 00 00 00 00 00 00 00 65 8D
// SHORT: dst = 0x00 00 00 00 00 00 00 00 00 00 00 00 14 11 40 51
// INT:   dst = 0x00 00 00 00 00 00 00 01 10 01 01 10 00 11 01
// LONG:  dst = 0x00 01 01 00 00 01 00 01 01 00 00 00 01 01 00 01
//
// The number of significant bits of src must be equal to lane_cnt. E.g., 0xFF658D which
// has 24 significant bits would be an invalid input if dst predicate register refers to
// a LONG type 1024-bit vector, which has at most 16 lanes.
void C2_MacroAssembler::sve_vmask_fromlong(PRegister dst, Register src, BasicType bt, int lane_cnt,
                                           FloatRegister vtmp1, FloatRegister vtmp2) {
  assert(UseSVE == 2 && VM_Version::supports_svebitperm() &&
         lane_cnt <= 64 && is_power_of_2(lane_cnt), "unsupported");
  Assembler::SIMD_RegVariant size = elemType_to_regVariant(bt);
  // Example:   src = 0x658D, bt = T_BYTE, size = B, lane_cnt = 16
  // Expected:  dst = 0b01101001 10001101
```

SVE的实现（~50行代码）

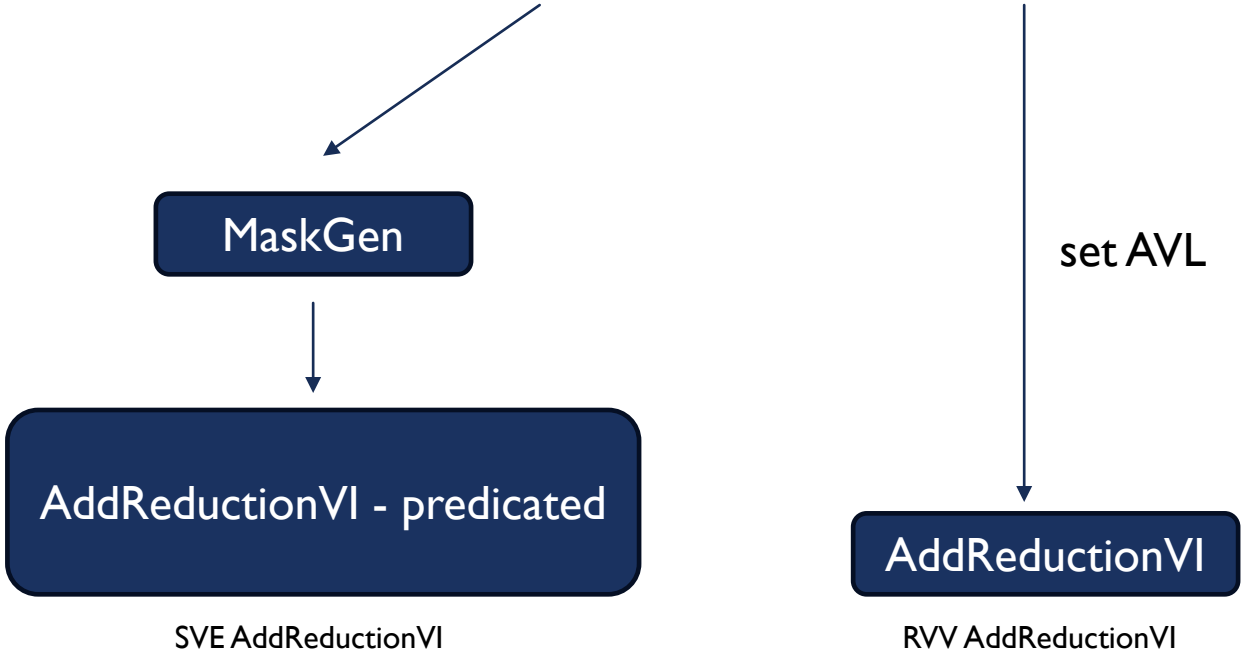
```
instruct vmask_fromlong(vRegMask dst, iRegL src) %{
  match(Set dst (VectorLongToMask src));
  format %{ "vmask_fromlong $dst, $src" %}
  ins_encode %{
    assert(Matcher::vector_length(this) <= XLEN, "precondition");
    __ vsetvli_helper(T_LONG, 1);
    __ vmv_s_x(as_VectorRegister($dst$$reg), $src$$Register);
  %}
  ins_pipe(pipe_slow);
%}
```

RVV的实现（2条指令）

OpenJDK Vector API 实现示例

SVE与RVV的mask操作对比(实际向量长度小于寄存器宽度，以AddReductionVI为例)

```
Node* VectorNode::Ideal(PhaseGVN* phase, bool can_reshape) {
  if (Matcher::vector_needs_partial_operations(this, vect_type())) {
    return try_to_gen_masked_vector(phase, this, vect_type());
  }
  return nullptr;
}
```



OpenJDK Vector API 实现示例

非masked及masked实现的loadmask

```
// vector load mask

instruct vloadmask(vRegMask dst, vReg src) %{
  match(Set dst (VectorLoadMask src));
  format %{ "vloadmask $dst, $src" %}
  ins_encode %{
    __ vsetvli_helper(T_BOOLEAN, Matcher::vector_length(this));
    __ vmsne_vx(as_VectorRegister($dst$$reg), as_VectorRegister($src$$reg), zr);
  %}
  ins_pipe(pipe_slow);
%}

instruct vloadmask_masked(vRegMask dst, vReg src, vRegMask_V0 v0) %{
  match(Set dst (VectorLoadMask src v0));
  format %{ "vloadmask_masked $dst, $src, $v0" %}
  ins_encode %{
    __ vsetvli_helper(T_BOOLEAN, Matcher::vector_length(this));
    __ vmsne_vx(as_VectorRegister($dst$$reg), as_VectorRegister($src$$reg), zr, Assembler::v0_t);
  %}
  ins_pipe(pipe_slow);
%}
```

OpenJDK Vector API 实现示例

与Zbb结合使用

```
bool Matcher::match_rule_supported_vector(int opcode, int vlen, BasicType bt) {
  if (!UseRVV) {
    return false;
  }

  if (!match_rule_supported(opcode) || !vector_size_supported(bt, vlen)) {
    return false;
  }

  switch (opcode) {
    case Op_VectorMaskLastTrue:
      if (!UseZbb || vlen > XLEN) {
        return false;
      }
      break;
    case Op_VectorMaskToLong:
    case Op_VectorLongToMask:
      if (vlen > XLEN) {
        return false;
      }
      break;
    default:
      break;
  }
  return true;
}
```

RISC-V Port Supported vector match rule

```
instruct vmask_lasttrue(iRegINoSp dst, vRegMask src) %{
  match(Set dst (VectorMaskLastTrue src));
  format %{ "vmask_lasttrue $dst, $src" %}
  ins_encode %{
    uint vector_length = Matcher::vector_length(this, $src);
    assert(UseZbb && vector_length <= XLEN, "precondition");
    __ vsetvli_helper(T_LONG, 1);
    __ vmv_x_s($dst$$Register, as_VectorRegister($src$$reg));
    if (XLEN != vector_length) {
      __ slli($dst$$Register, $dst$$Register, XLEN - vector_length);
      __ srli($dst$$Register, $dst$$Register, XLEN - vector_length);
    }
    __ clz($dst$$Register, $dst$$Register);
    __ mv(t0, XLEN - 1);
    __ sub($dst$$Register, t0, $dst$$Register);
  %}
  ins_pipe(pipe_slow);
%}
```

VectorMaskLastTrue的C2 instruct

OpenJDK Vector API 在RISC-V平台的未来工作

- 优化冗余的vset指令
- 持续跟进上游优化及JEP
- RVV1.0的板子上进行Benchmark及JMH测试

OpenJDK Vector API 在RISC-V上的实现

谢谢大家！