RISC-V Vector ISA Spec —

Keystone of RISC-V Based High Performance Computing

Stream Computing Inc.

Mark Zhan

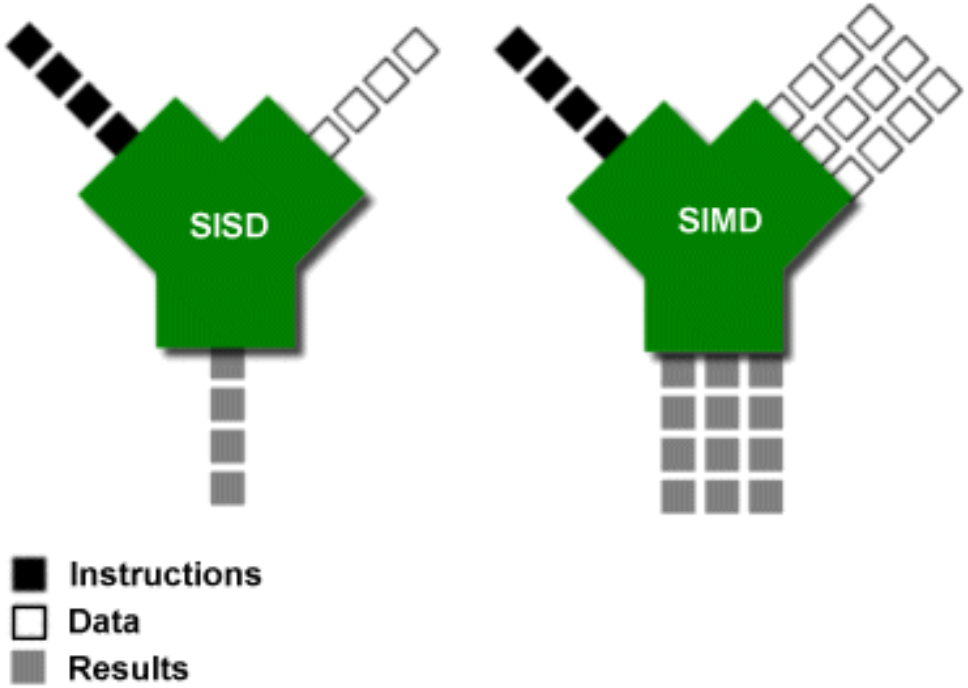
OSDT2019

SIMD Instructions

- Parallelism Technology of Modern HPC Processor:
  - **Instruction Stream**: *Instruction Level Parallelism (ILP)* —— VLIW, Superscalar
  - **Data Stream**: *Data Level Parallelism —— SIMD*

- SIMD-based Data Parallelism: when you have a large mass of data of a uniform data-type that needs the same instruction performed on it.
  - Matrix Computing: Image Processing, Machine Learning, scientific computation etc.

- SIMD Instruction Programming Model:
  - Pure SIMD machine (e.g: SIMD-only CPU)? NO!
  - As a part of a SISD host machine. Programs are written for a SISD machine, and include in their code SIMD instructions.

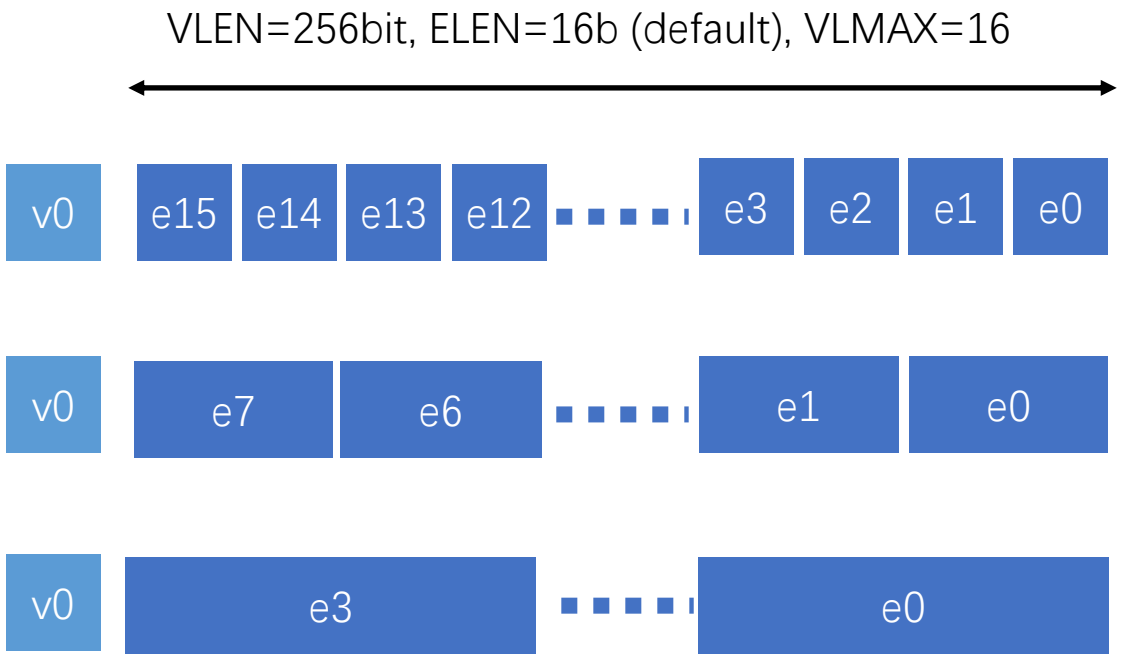


RISC-V SIMD ISA

- RISC-V SIMD ISA: V-Extension (aka: V-Spec) vs. P-Extension
- V-Spec ISA:
  - Started at 2015 as a WG under RISC-V Foundation Org, lead by Krste Asanović from UC Berkeley
  - Current Status: version 0.7— stable enough proposal, to be submitted to RISC-V Foundation, base of toolchain support & functional simulators implementation.
  - Home Page: https://github.com/riscv/riscv-v-spec
  - Target Application Domains: Machine Learning, DSP, Crypto, Numerical Computation etc.
- P-Extension:
  - Donated By Andes Technology
  - Coming from Andes DSP SIMD ISA

V-Spec ISA— Hart Implementation Independent Design

- Bit-width of Vector Register is up-to the hart implementation for silicon vendor.
  - ***VLEN***: the max bit-width of vector register in a specific hart
- Dynamic Configurable Vector Element Width: 8、16、32…1024 bit vector element are supported —— max(XLEN, FLEN):
  - ***ELEN***: The max element bit-width allowed in a specific hart implementation
  - ***SEW***: Standard Element Width — Define the current vector element width in current instruction context, dynamically configured by vsew[2:0] in vtype CSR.
- Dynamic Register Grouping
  - group 2/4/8 vector registers to a wider vector register, to allow a SIMD insn to operate larger vector.
  - ***LMUL***: Define how many vector registers to form a register group
  - ***VLMAX*** = LMUL * (VLEN/SEW)

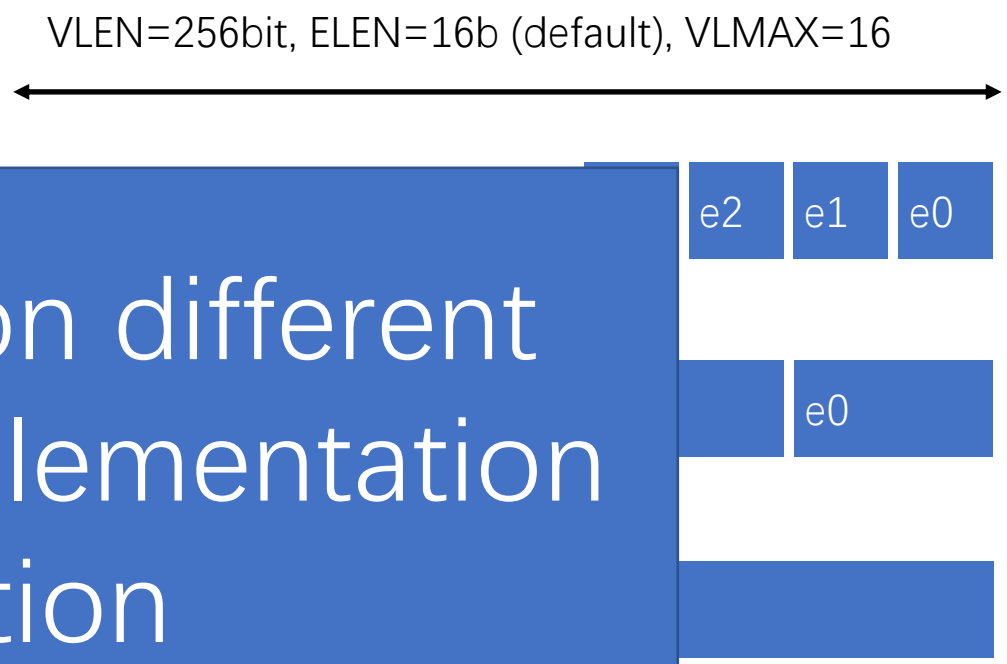


- Dynamic AVL (Application Vector Length): configured by ***vl*** CSR
  - AVL define how many vector elements are operated by current SIMD Vector Instruction

V-Spec ISA— Hart Implementation Independent Design

- Bit-width of Vector Register is up-to the hart implementation for silicon vendor.
  - ***VLEN***: the max bit-width of vector register in a specific hart
- Dyna
  32…1
  max(
  - ***EL***
    im
  - ***SE***
    ele
    co
- Dyna
  - group 2/4/8 vector registers to a wider vector register, to allow a SIMD insn to operate larger vector.
  - ***LMUL***: Define how many vector registers to form a register group
  - ***VLMAX*** = LMUL * (VLEN/SEW)

Same binary can run on different V-Spec Hart silicon implementation w/o recompilation



- Dynamic AVL (Application Vector Length): configured by ***vl*** CSR
  - AVL define how many vector elements are operated by current SIMD Vector Instruction

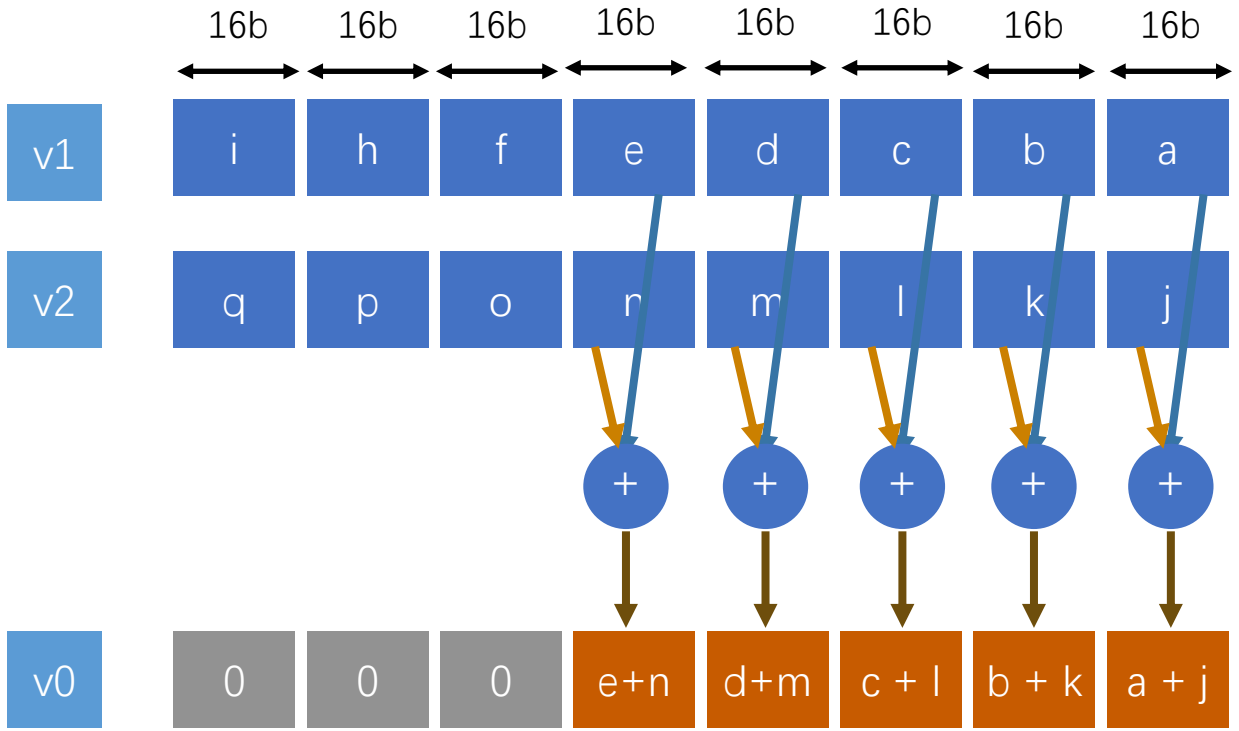
Example — How is this executed?

Example: Add two Arrays

```
// Pseudo Codes
for (i = 0; i < 5; i++)
{
    v0[i] = v1[i] + v2[i];
}
for (i = 5; i < MAXVL; i++)
{
    v0[i] = 0;
}
```

Vectorize

```
VL=5, MAXVL=8
# Assume GPR t0 = 5
vsetvli x0, t0, e16 # set vl = 5
vadd.vv v0, v1, v2   # v0 = v1+v2
```

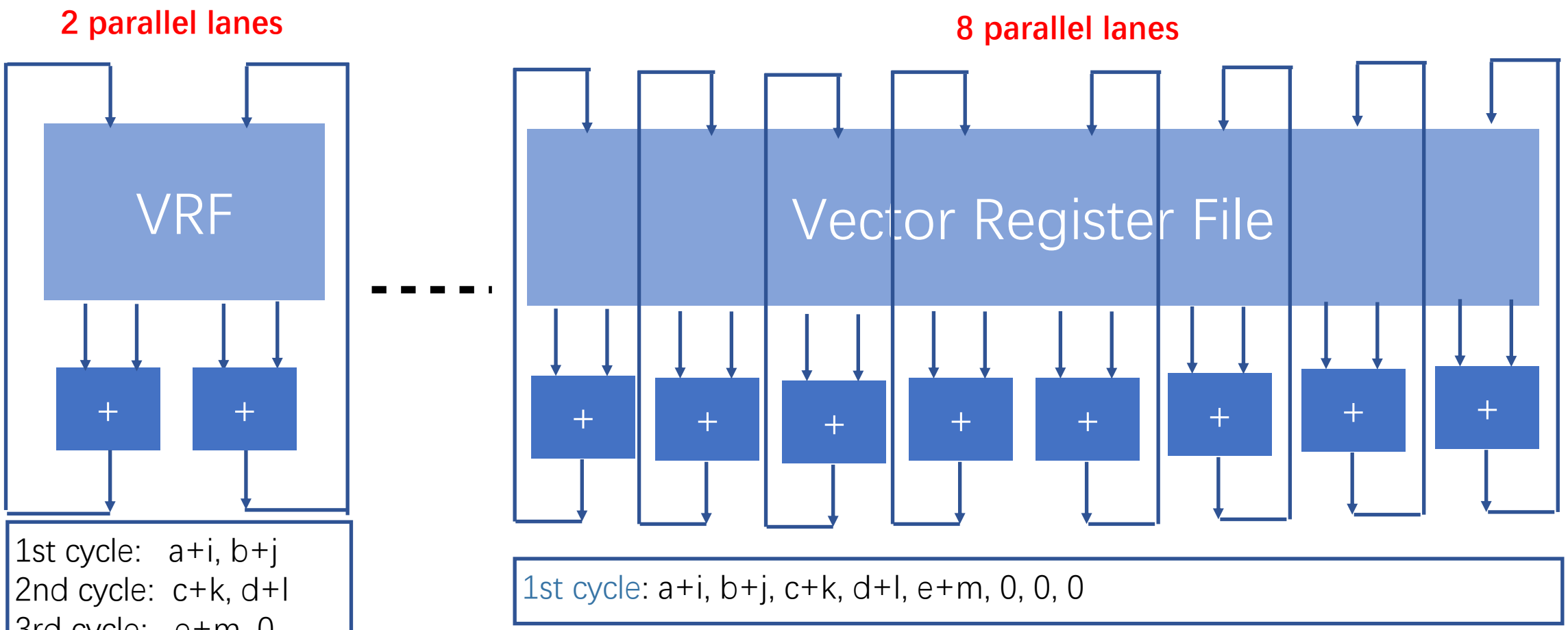


Parallel execute op to each vector element

SIMD Instruction Parallelism

Example — How is this executed?

- The # of Parallel lanes is up to silicon vendor implementation!
- RISC-V Vector ISA: The # of Parallel lanes is transparent to software. Same code runs independent of # of lanes

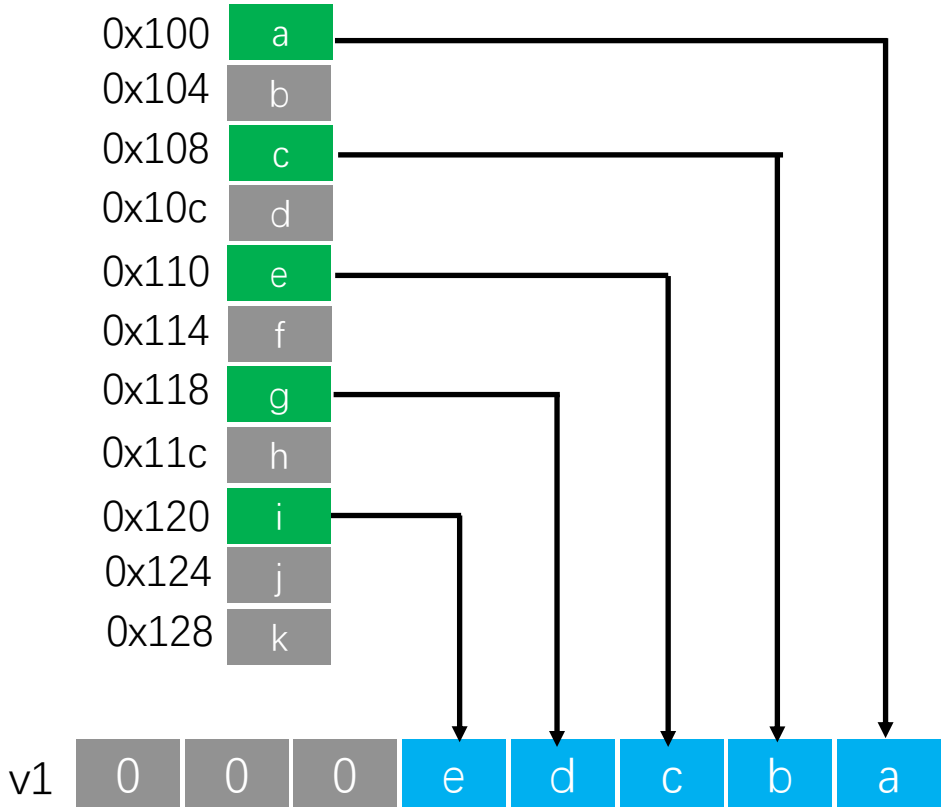


V-Spec ISA Overview

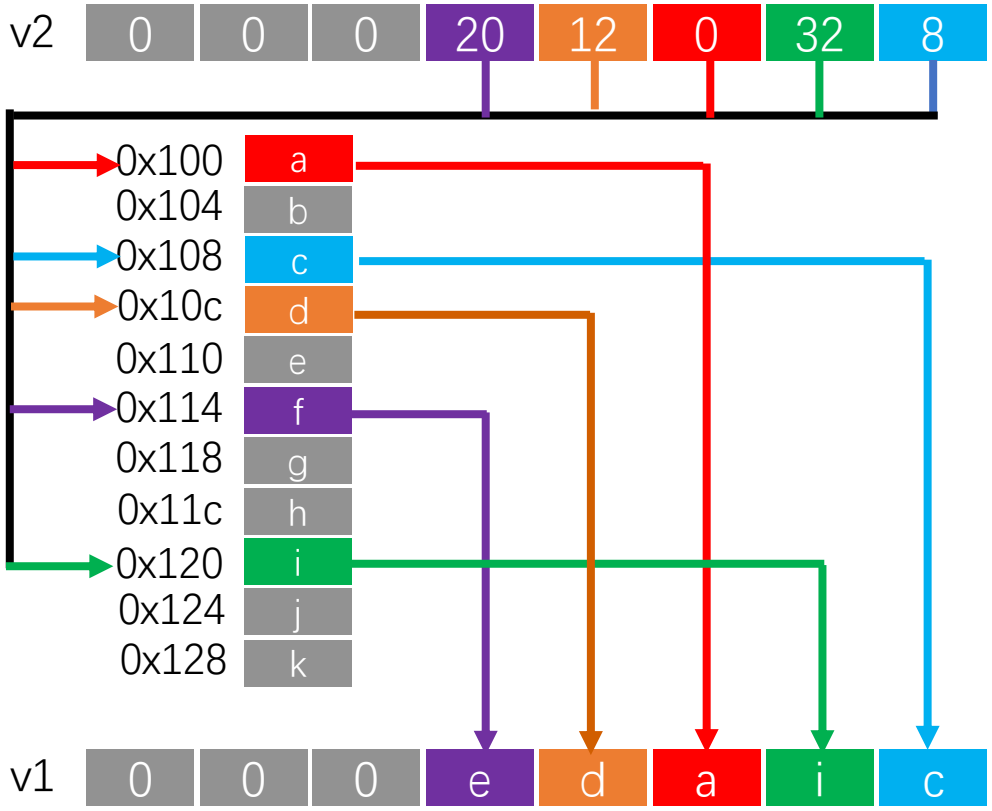
Category	Vector Instruction	Major OPCODE	Function Desc
Vector Memory Inst	vl - vector load	LOAD-FP (0000111)	Move data between vector registers and memory
	vs - vector store	STORE-FP (0100111)	
Vector AMO Inst	vamoswap/vamoadd/vamoxor/vamoand/vamoor/vamomin/vamomax/vamominu/vamomaxu	AMO (0101111)	Atomic Memory Operations. Performs an atomic read-modify-write to each vector element in memory. Usually supported by UNIX-Profile
Vector Arithmetic Inst	vector arithmetic Inst.	OP-V (1010111)	Various Vector Element Data Type: - Integer: - Fixed-Point - Floating-point
	vector reduction inst.		
	vector mask inst.		
	vector permutation inst.		
			Arithmetic Operations: - add/substract - mul/fused multiply-add - logical inst. - min/max inst. - vector merge - vector square root
Vector CSR R/W Inst.	vsetvl/vsetvli	OP-V (1010111)	read/write Vector CSR vtype & vl

Vector Load/Store Instructions Addressing Modes

1. **Unit-stride Mode**: read/write values in continuous memory specified by (rs1) base address.
2. **Strided Mode**: vd[i] = base address (rs1) + byte offset (rs2) * (i-1)
3. **Indexed Mode (scatter/gather)**: byte offset specified by the element value of vector offset operand "**vs2**"



Example: vlsw.v v1, (rs1), **rs2**



Example: vlxw.v v1, (rs1), v2

V-Spec Vector Arithmetic Instruction Features

- Data Types:
  - Integer
  - Fixed-Point
  - Floating-Point
- Widening operations
  - Destination elements are 2*SEW wide
  - written with a vw* prefix on the opcode or vfw* for vector floating-point operations
- Narrowing operations
  - written with a vn* prefix on the opcode or vfn* for vector floating-point operations
- Source Operands:
  - Vector-Vector
  - Vector-Scalar
  - Vector-Immediate
- Vector DataType Convert Instructions
  - E.g: Floating-Point ⇔ Integer
- Vector Mask Instructions are supported:
  - least-significant bit of vector element is used as mask bit
  - v0 register is always used as source mask register

V-Spec Vector Arithmetic Instruction Features

- Arithmetic Instructions Categories:
  - Add/Substract, Multiply/Divide
  - FMA: Fused Multiply-Add
  - Logical Instructions: AND, OR, XOR, CMP, Shift etc
  - MIN/MAX Instructions
  - Merge Instruction
  - Integer/Floating-Point Reduction Instructions
  - Permutation Instructions: slideup, slidedown etc

V-Spec ISA vs Others

- RISC-V V-Spec is the best Mixed-Precision Vector ISA. Other Commercial Vector ISA (such as: X86 AVX, ARM Neon, etc) has certain architectural shortcomings:
  - ***Opcodes designate a fixed vector length***. Exposing the hardware SIMD width in the ISA hinders portability. Code must be recompiled to benefit from longer SIMD widths
  - ***All vector registers are uniformly fixed in size***. Consequently, the number of elements per vector register varies at different precisions, and it is not so straightforward to chain mixed- precision operations.
  - ***The vector length is typically short***, currently at most 256 bits. These SIMD extensions, electing for the path of least re-sistance towards subword parallelism, re-use existing scalar datapaths and control. With shorter hardware vectors, more stripmine iterations are required.

V-Spec ISA S/W Community Status

- V-Spec ISA Software Support Status:
  - Spike Simulator is released
  - Binutils is released
- Call for Community Participation for:
  - GCC/LLVM Compiler support
  - BLAS Library support
  - Gem5 simulator support
  - etc.

Thanks.