RISC-V Vector Extension Webinar II

August 3th, 2021

Thang Tran, Ph.D.
Principal Engineer

Webinar II - Agenda

- Andes overview
- Vector technology background
  - SIMD/vector concept
  - Vector processor basic
- **RISC-V V extension ISA**
  - **Basic**
  - **CSR**
- RISC-V V extension ISA
  - Memory operations
  - Compute instructions
- Sample codes
  - Matrix multiplication
  - Loads with RVV versions 0.8 and 1.0
- AndesCore™ NX27V
- Summary

Copyright© 2020 Andes Technology Corp.

Terminology

- ISA: Instruction Set Architecture
- GOPS: Giga Operations Per Second
- GFLOPS: Giga Floating-Point OPS
- **XRF**: Integer register file
- FRF: Floating-point register file
- **VRF**: Vector register file
- SIMD: Single Instruction Multiple Data
- MMX: Multi Media Extension
- SSE: Streaming SIMD Extension
- AVX: Advanced Vector Extension
- ACE: Andes Custom Extension
- CSR: Control and Status Register
- **SEW**: Element Width (8-64)
- ELEN: Largest Element Width (32 or 64)
- **XLEN**: Scalar register length in bits (64)
- FLEN: FP register length in bits (16-64)
- **VLEN**: Vector register length in bits (128-512)
- **LMUL**: Register grouping multiple (1/8-8)
- EMUL: Effective LMUL
- **VLMAX**/MVL: Vector Length Max
- AVL/**VL**: Application Vector Length
- **Configurable**: parameters are fixed at built time, i.e. cache size
- **Extensible**: added instructions to ISA includes custom instructions to be added by customer
- **Standard extension**: the reserved codes in the ISA for special purposes, i.e. FP, DSP, …
- **Programmable**: parameters can be dynamically changed in the program

Copyright© 2020 Andes Technology Corp.

RISC-V V Extension ISA Basic

Copyright© 2020 Andes Technology Corp.

Vector Register ISA

- *Vector-Register ISA Definition*:
  - All **vector operations are between vector registers** (except for load and store). This is in contrast with ***Vector-Memory ISA*** where all vector operations are on data memory
  - Vector load-store unit fetches and stores the memory data to/from vector registers
- Basic requirements
  - Each vector data register hold **N M-bit** values
    - SEW – single vector element in bits, **M-bit**, power of 2
    - VLEN – number of bits in a vector register, **N*M-bit**, power of 2, VLEN > SEW
    - VLMAX – number of vector elements, **N**, in the vector register, **VLEN/SEW**
    - Elements are independently executed (**mostly, except for permute instruction**)
  - Vector control registers: VMASK (mask – predication – valid element for execution) is **v0**
  - Vector instructions
    - Arithmetic, logical, shift, multiply, divide, **mask**, permute, floating-point functional units
    - Vector load/store unit: load/store with stride, load/store with index, segment load/store

Copyright© 2020 Andes Technology Corp.

Vector Processing ISA Basic

Scalar	**Standard scalar RISC-V ISA**					
Scalar FP	**Standard floating-point RISC-V ISA**					
Vector ALU	ALU Mult Macc Mask Permute Divide	s.int u.int fp	8 16 32 64	.vv .vx .vi	Saturate Overflow	masked unmasked
Vector Memory	load store	s.int u.int	4 8 16 32 64	8 16 32 64	constant indexed segment	masked unmasked
Vector registers	**32x(128b, 256b, 512b)**					

Custom Int4

Include custom Bfloat16

Andes Vector processor: VLEN=512b, SEW=8,16,32,64b, elements=64,32,16,8

These numbers are used for the rest of the presentation

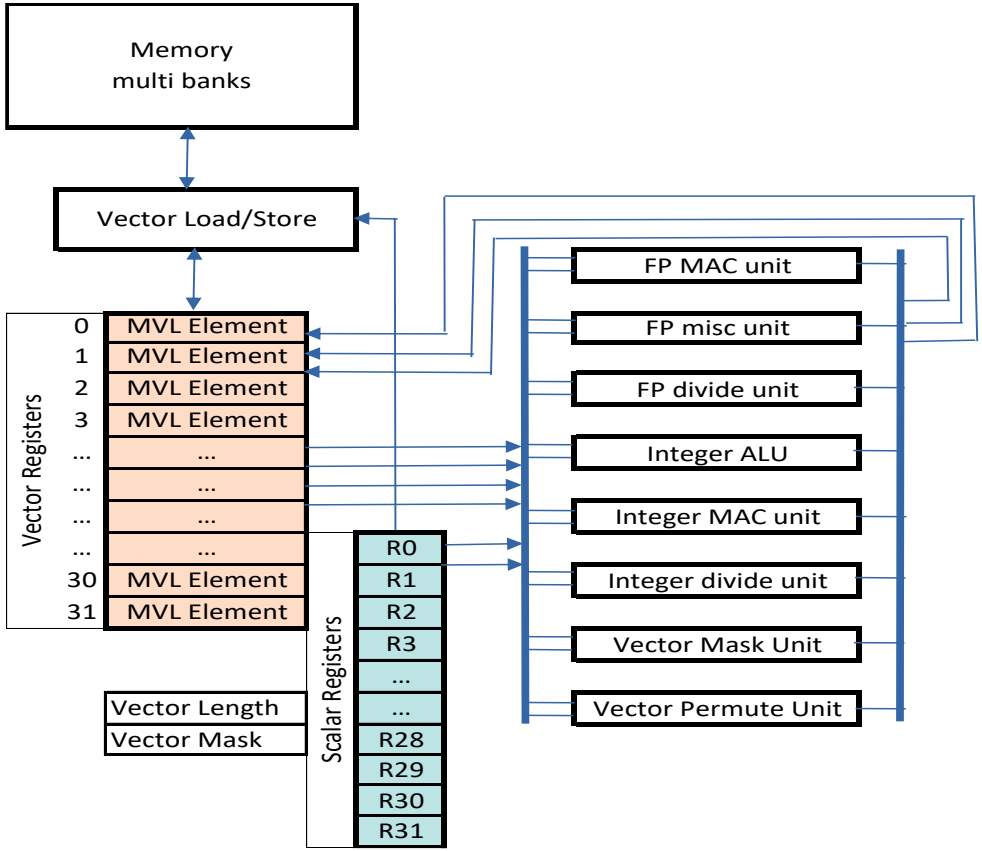
Copyright© 2020 Andes Technology Corp.

Notations

- Register references
  - vs1, vs2, vd – source and destination vector registers, VRF
  - rs1, rs2, rd – source and destination integer registers, XRF
  - fs1, fs2, fd – source and destination floating point registers, FRF
- Register numbers
  - v0, v1, …, v30, v31 – vector register numbers (addresses, indices)
  - x0, x1, …, x30, x31 – integer register numbers (addresses, indices)
  - f0, f1, …, f30, f31 – floating-point register numbers (addresses, indices)
  - **x0** is not real register; source register data = 0, destination register = no write back
  - **v0** can be used for vector mask and carry bits

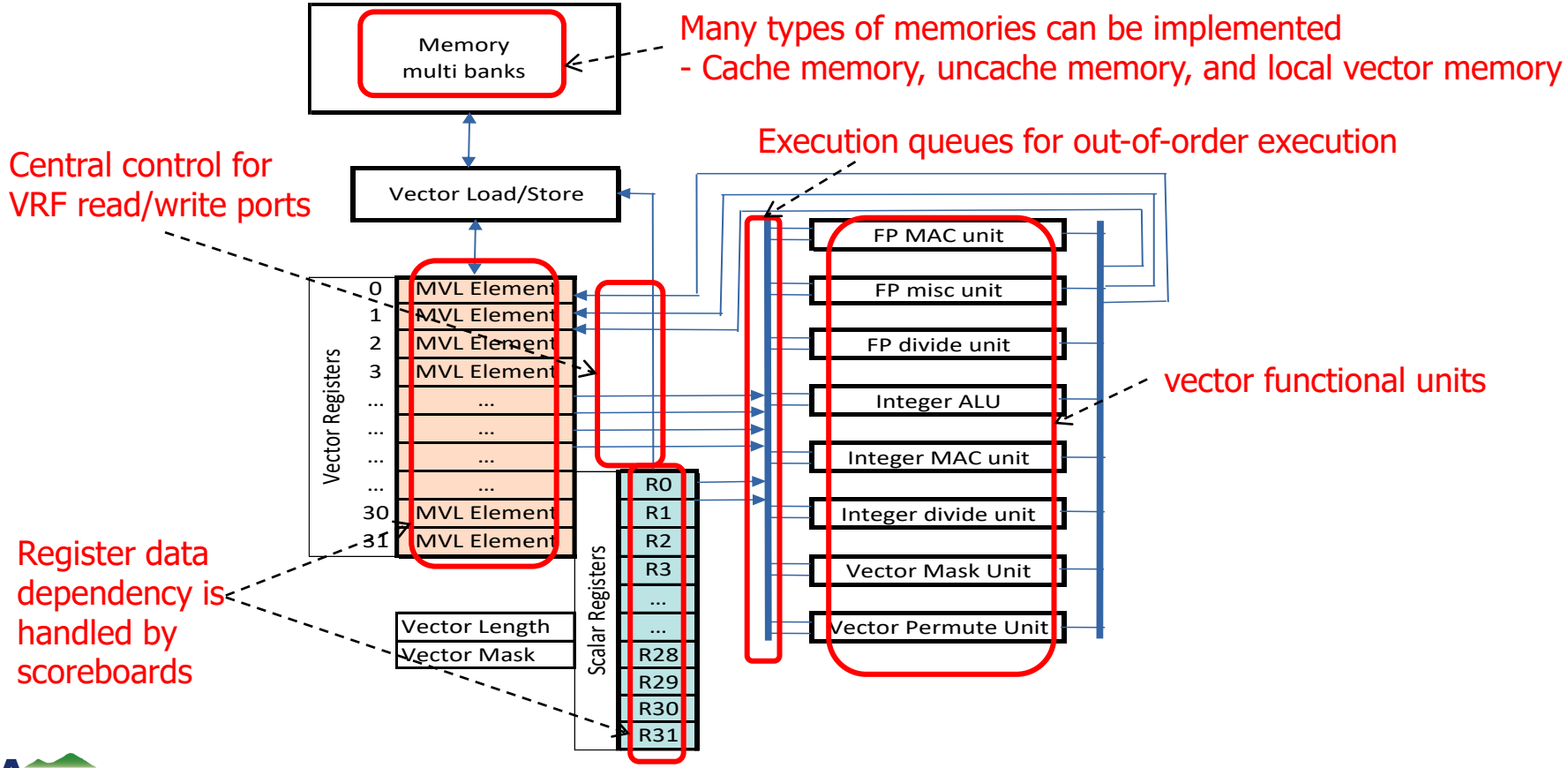
Copyright© 2020 Andes Technology Corp.

Vector Processor Overview - Block Diagram



Copyright© 2020 Andes Technology Corp.

Vector Processor Microarchitecture Features



Copyright© 2020 Andes Technology Corp.

RISC-V V Extension ISA

CSR

Copyright© 2020 Andes Technology Corp.

Vector CSR Registers

Address	Privilege	Name	Description
0x008	URW	vstart	Vector start position
0x009	URW	vxsat	Fixed-Point Saturate Flag
0x00A	URW	vxrm	Fixed-Point Rounding Mode
0x00F	URW	vcsr	Vector control and status register
0xC20	**URO***	**vl**	Vector length
0xC21	**URO***	**vtype**	Vector data type register
0xC22	URO	vlenb	VLEN/8 (vector register length in bytes) – configuration register

*written by **vsetvl(i)** instruction, not by CSR access instructions

Copyright© 2020 Andes Technology Corp.

CSR – Vector Execution Fixed Point Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved																															sat

- vxsat – Vector Fixed-Point Saturation Flag Register – single bit to indicate that saturated value was written to destination vector register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved																														vxrm	

- vxrm – Vector Fixed-Point Rounding Mode Register:
  - 00: rnu – round to nearest up (add +0.5 to LSB)
  - 01: rne – round to nearest even
  - 10: rdn – round down (truncate)
  - 11: rod – round to odd (OR bits into LSB, “jam”)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved																													vxrm		sat

- vcsr – merging of vxsat and vxrm into single CSR, explicitly reset by software

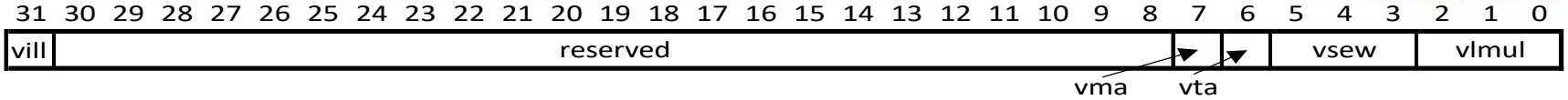
Copyright© 2020 Andes Technology Corp.

CSR – Vector Execution Floating-Point Register

- The vector processor uses the fflags and frm from the FPU
  - fflags[4:0]: floating-point accrued exception flags
  - frm[2:0]: floating-point rounding modes

Copyright© 2020 Andes Technology Corp.

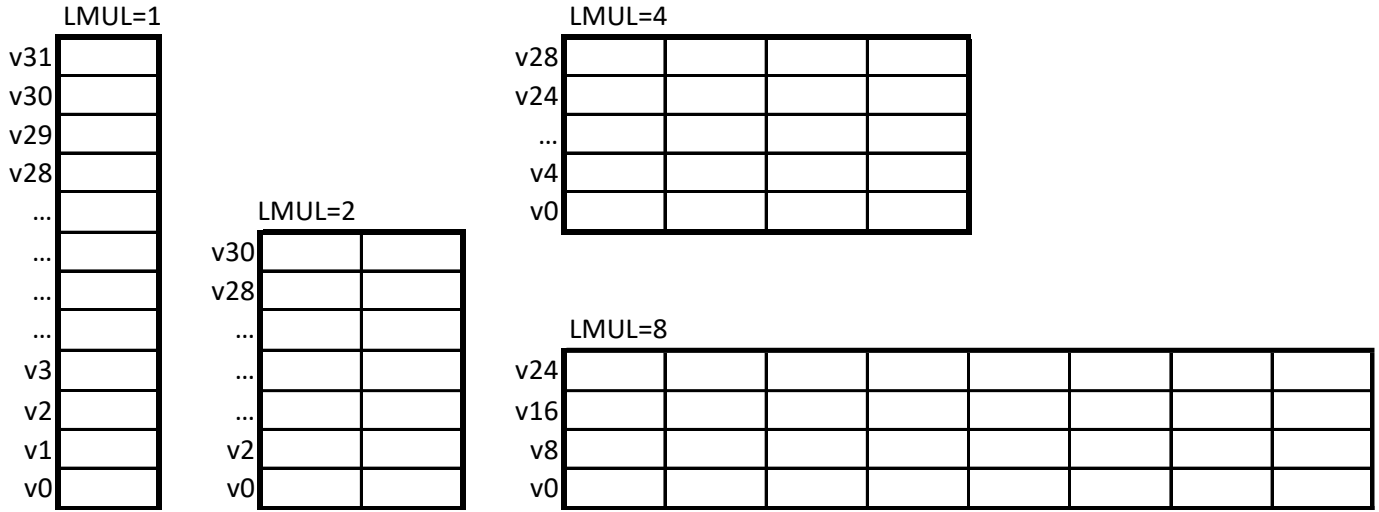
CSR – Vector Type Register (vtype)



- vlmul[2:0] – **grouping of vector registers**, 2x, 4x, 8x, or fraction of vector register, 1/2, 1/4, 1/8
  - 000: no grouping, 32 vector registers, VLMAX = VLEN/SEW
  - 001: 16 vector registers, effective VLEN is **2X**, twice number of elements
  - 010: 8 vector registers, effective VLEN is **4X**, quadruple number of elements
  - 011: 4 vector registers, effective VLEN is **8X**, octuple number of elements
  - 100: Reserved
  - 101: Fractional vector register, effective VLEN is **1/8**, one-eighth number of elements
  - 110: Fractional vector register, effective VLEN is **1/4**, quarter number of elements
  - 111: Fractional vector register, effective VLEN is **1/2**, half number of elements
- vsew[2:0] – vector standard element width, VLMAX = VLEN/VSEW
  - 000 – 011: **8, 16, 32, 64,** (100-111: reserved for larger sizes of 128, 256, 512, 1024)
- vta – vector tail agnostic, implementation dependent, should the data is unknown or undisturbed
- vma – vector mask agnostic, implementation dependent, should the data is unknown or undisturbed
- vill – illegal value in the vsetvl(i) instruction, any subsequent vector instruction is illegal instruction

Copyright© 2020 Andes Technology Corp.

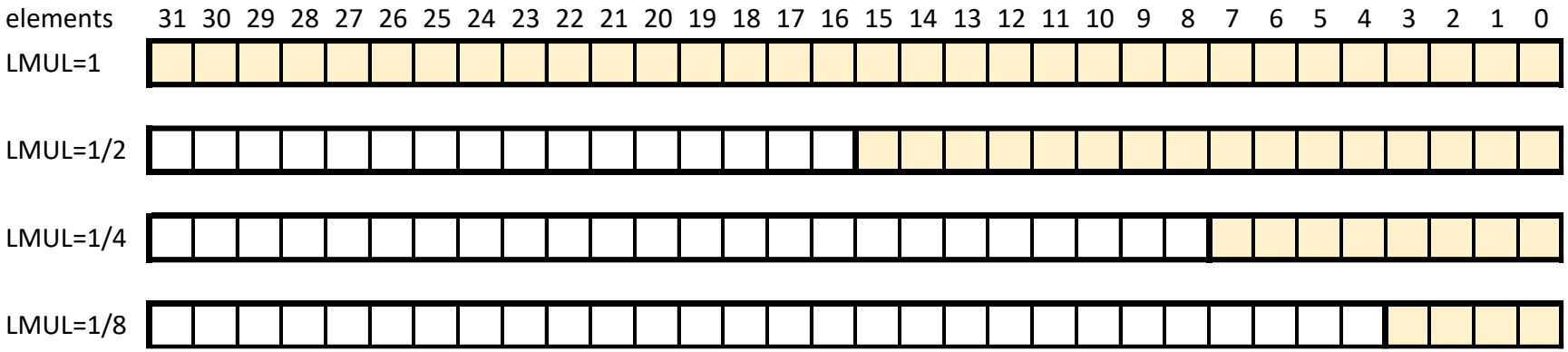
LMUL – Registers Grouping



- LMUL=2, instructions with odd register are illegal instructions
- LMUL=4, vector registers are incremented by 4, else illegal instructions
- LMUL=8, only v0, v8, v16, and v24 are valid vector registers

Copyright© 2020 Andes Technology Corp.

LMUL – Fractional Registers

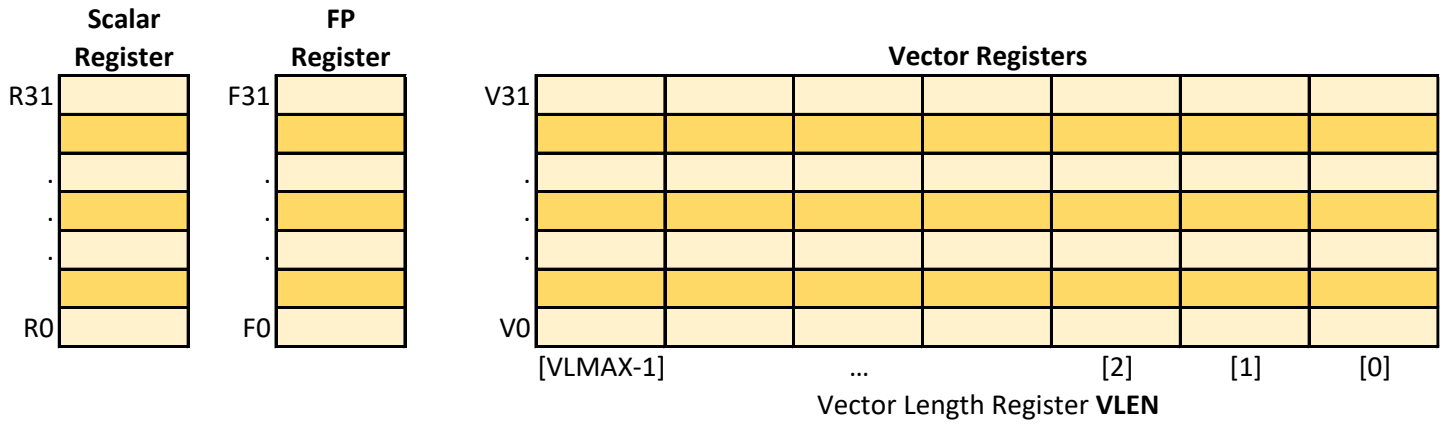


- LMUL=1/2, load data into half vector register, then perform vector double operations to expand data to the whole register
- LMUL=1/4, using vector extension of 4x to expand data to the whole register
- LMUL=1/8, using vector extension of 8x to expand data to the whole register

Copyright© 2020 Andes Technology Corp.

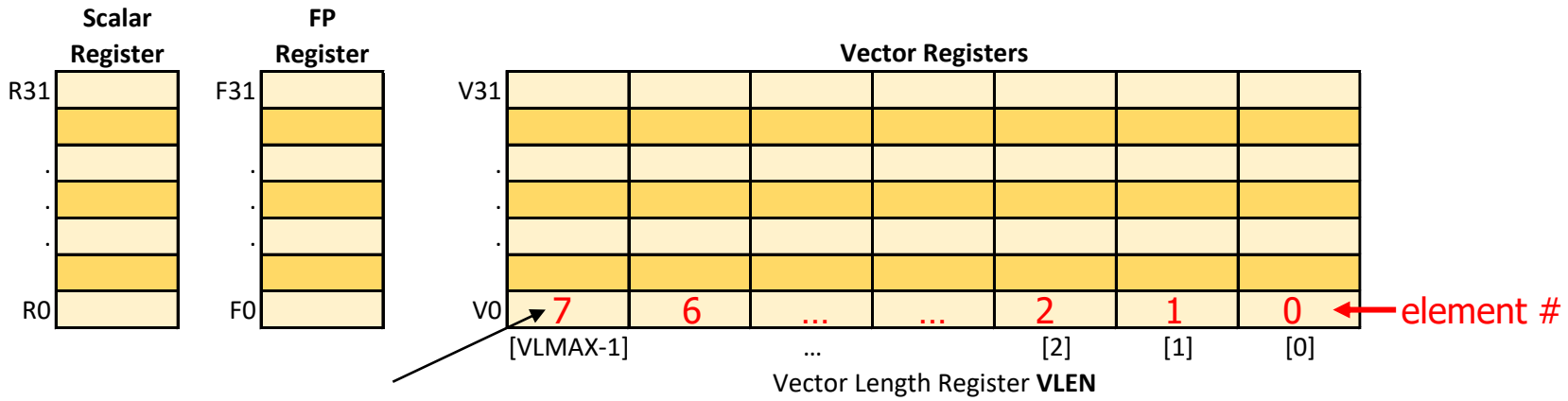
Vector Register Programming Model

- 32 vector registers – 5-bit
- VLEN = The width of each vector register[i] in power of 2
- Version 0.7: FPU cannot shares the vector registers



Copyright© 2020 Andes Technology Corp.

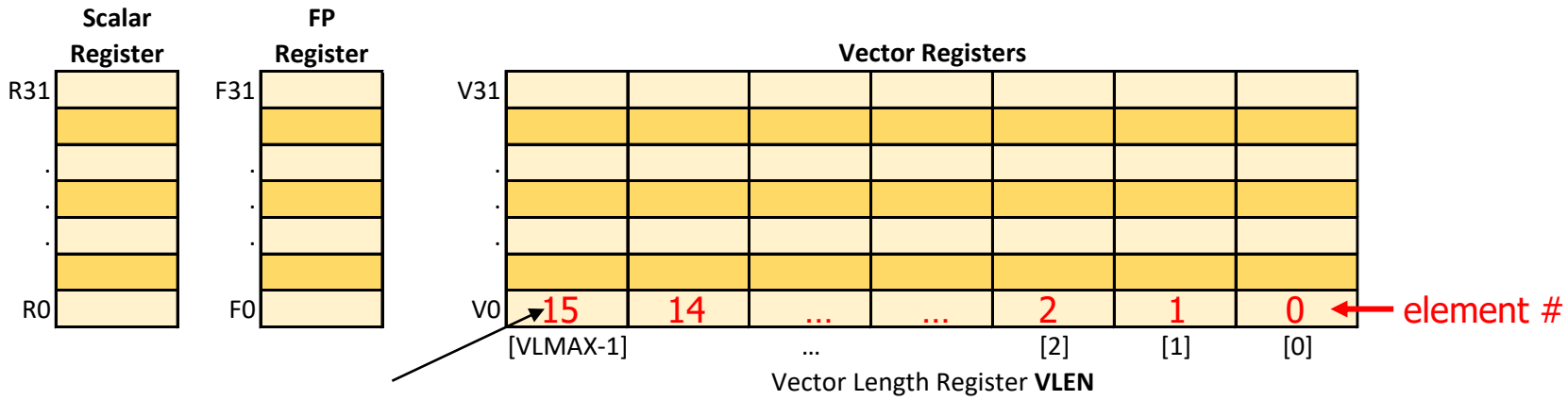
Vector Register 64-bit Elements



VLEN=512b, SEW=64b

Copyright© 2020 Andes Technology Corp.

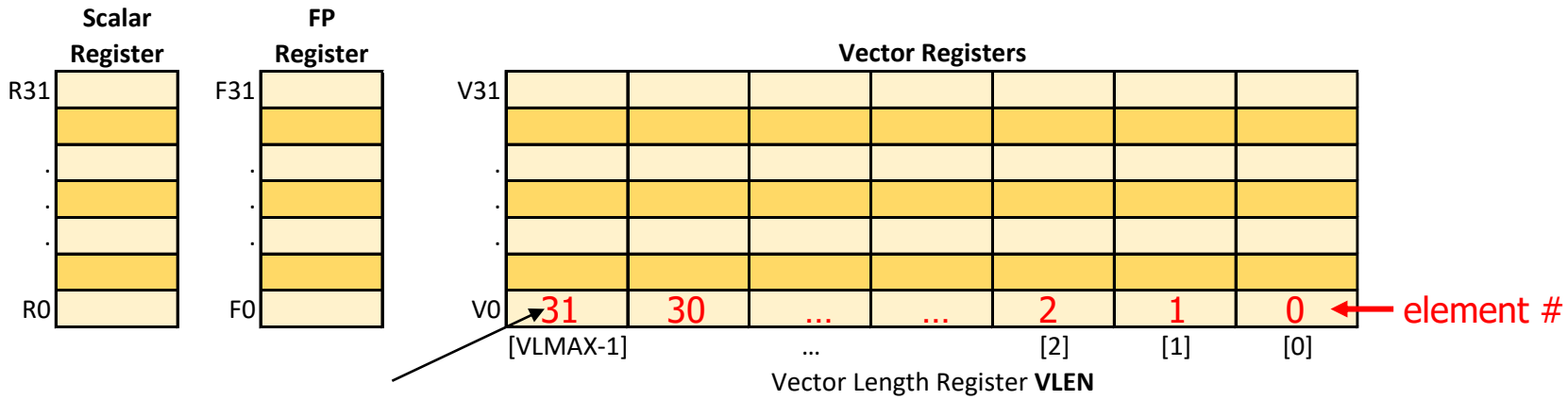
Vector Register 32-bit Elements



VLEN=512b, SEW=32b

Copyright© 2020 Andes Technology Corp.

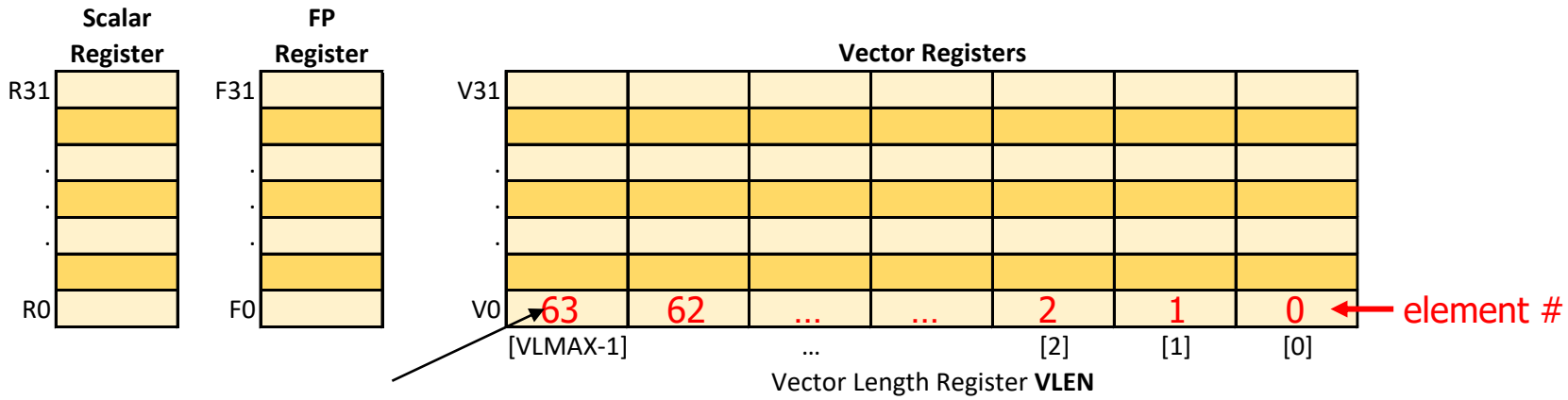
Vector Register 16-bit Elements



VLEN=512b, SEW=16b

Copyright© 2020 Andes Technology Corp.

Vector Register 8-bit Elements



VLEN=512b, SEW=8b

Copyright© 2020 Andes Technology Corp.

Element Numbering, LMUL=2

- Straight forward numbering, increment sequentially
- The numbering is important for mask and permutation operations

Vm*n	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	SEW=8
Vm*n+1	127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96	95	94	93	92	91	90	89	88	87	86	85	84	83	82	81	80	79	78	77	76	75	74	73	72	71	70	69	68	67	66	65	64	

Vm*n	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	SEW=16
Vm*n+1	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32	

Vm*n	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	SEW=32
Vm*n+1	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	

Vm*n	7	6	5	4	3	2	1	0	SEW=64
Vm*n+1	15	14	13	12	11	10	9	8	

Copyright© 2020 Andes Technology Corp.

Vector Register 8-bit Elements, LMUL

Vector Registers

V31							
.							
.							
V8							
V7	511	510	…	…	…	449	448
V6	447	446	…	…	…	385	384
V5	383	382	…	…	…	321	320
V4	319	318	…	…	…	257	256
V3	255	254	…	…	…	193	192
V2	191	190	…	…	…	129	128
V1	127	126	…	…	…	65	64
V0	63	62	…	…	…	1	0
	[VLMAX-1]		…		[2]	[1]	[0]

V0, LMUL=8

V0, LMUL=4

V0, LMUL=2

element #

Vector Length Register **VLEN**

VLEN=512b, SEW=8b

Copyright© 2020 Andes Technology Corp.

LMUL=8 and Chaining Example

Cycles:	1	2	3	4	5	6	7	8	9	10
vmul v0, v8, v16	X1	X2								
v1, v9, v17		X1	X2							
……			…	…	…	…				
v7, v15, v23										
vadd v24, v0, v8			X1							
v25, v1, v9				X1						
……				…	…	…	…	…	…	…

Vector length: VLEN * LMUL
- LMUL=1 → 512 bits
- LMUL=8 → 4096 bits

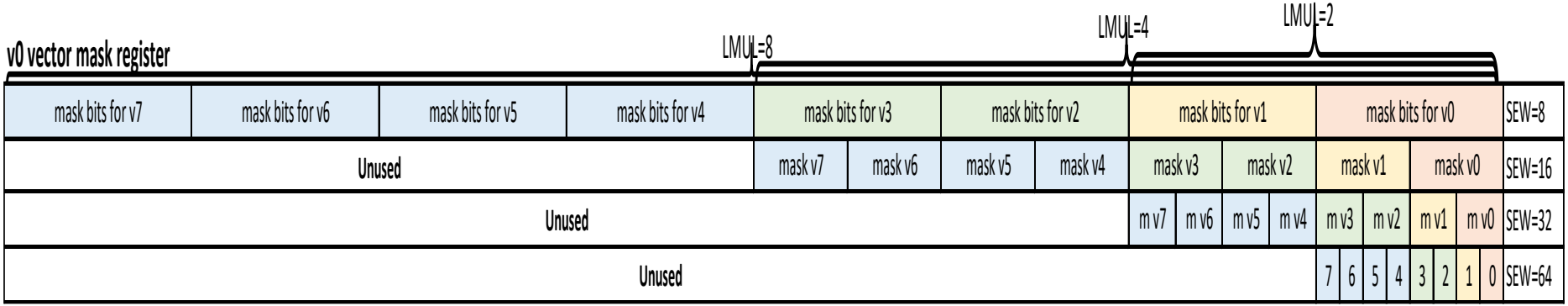
V0~V7 group 1
V8~V15 group 2
V16~V23 group 3
V24~V31 group 4

Xn: execution LMUL → Chaining →

Copyright© 2020 Andes Technology Corp.

Vector Mask – Conditional or Predicate Execution

- v0 is used for vector mask, **single mask for all LMUL values**
  - 1 mask bit for each element
    - 0 – mask (mask-off), no write to destination vector elements, cannot have exception
    - 1 – unmask
  - The mask is enabled for each vector instruction with inst[25]=0, inst[25]=1 is unmask
    - vop.v v8, v16, 3, v0.t // mask is enable, each element writes back with v0_mask[i]=1
    - vop.v v8, v16, 3 // unmask, all elements write result data back to VRF
- Note: new vector mask load/store maybe added



Copyright© 2020 Andes Technology Corp.

CSR - Vector Length Register (vl)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved																						vl[VLMAX*8-1:0]									

- vl is the number of elements for vector operations
  - Updated by vsetvl(i) instruction along with the vtype CSR
    - vl is set to VLMAX if the set value is greater than VLMAX
    - The default vl is VLMAX
    - If x0 is used to set vl, then vl is set to VLMAX
  - Force to an element by fault-only-first load instruction
  - vl=0, then no elements are updated in vector operations

elements	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
vl=VLMAX																																
vl=21																																

Copyright© 2020 Andes Technology Corp.

CSR Vector Instruction – vsetvli/vsetvl

- Two vector CSRs are updated by single instruction
  - vsetvli rd, rs1, vtypei // vl=rs1 (which is also AVL, application vector length), rd=AVL
  - vsetvl rd, rs1, rs2 // rd = avl
- CSR updates:
  - csr.vtype = vtypei or rs2
    - {m1}, m2, m4, m8, mf2, mf4, mf8 – setting the LMUL, default is m1 if m value is not specified
    - e8, e16, e32, e64 – setting the SEW size
    - {tu}, ta – setting tail elements, default is tu if ta is not specified
    - {mu}, ma – setting masked-off elements, default is mu if ma is not specified
  - csr.vl = rs1
    - rd=rs1=x0 no update for csr.vl
    - rd!=x0, rs1=x0 vl=VLMAX, x[rd]=vl
    - rs1!=x0 vl=x[rs1], x[rd]=vl
    - x[rs1] > VLMAX vl=VLMAX, x[rd]=vl

Copyright© 2020 Andes Technology Corp.

Sample Codes for LMUL and SEW Manipulation

- VLEN=512b, e8 to e32 elements for 16 elements

vl=16
sew=8

sew=8b
lmul=1/4

sew=32b
lmul=1

```
vsetvli t0, zero, e8, mf4      // set SEW=8b, LMUL=1/4, VLMAX=64/4=16 elements
vle8.v  v8, (a2)               // load 16 8b elements
vsext.vf4  v8, v8              // Signed extended 16 8b elements to 32b elements
vsetvli t0, zero, e32          // set SEW=32b, LMUL=1 (default), vl=16 (VLMAX)
vadd.vx v8, v8, t1             // 32-bit vector add for 16 32b elements
…
```

- Notes:
  - Write and read to CSRs are normally serial events but for VPU a speculative value of LMUL, SEW, and vl are used for subsequent instructions until the values are changed again

Copyright© 2020 Andes Technology Corp.

Optimized codes for Load and Signed Extension

- VLEN=512b, e8 to e32 elements for 16 elements

```
vsetvli t0, zero, e32        // set SEW=32b, LMUL=1, vl=512/32=16 elements
vle8.v  v8, (a2)             // load 16 8b elements, vle16.v or vle32.v is always 16 elements
vsext.vf4  v8, v8            // Signed extended 16 8b elements to 32b elements
vadd.vx v8, v8, t1           // 32-bit vector add for 16 32b elements
…
```

- Notes:
  - *In earlier version, 0.8, of RVV ISA, the vle8.v and vsext.vf4 instructions are replace by a single instruction, (the ACE load instruction can still do this load):*

```
vlb.v  v8, (a2),             // load 8b elements and signed extended to 32b elements, still 16 elements
```

Copyright© 2020 Andes Technology Corp.

Sample Code for Large Data Set

- VLEN=512b, e8 to e32 elements for 128 elements

```
vsetvli t0, zero, e32, m8      // set SEW=32b, LMUL=8, vl=512b*8/32b=128 elements
vle8.v  v6, (a2)               // load 128 8b elements into 2 vector registers, v6-v7
vsext.vf4  v0, v6              // Signed extended 128 8b elements to 32b elements
vadd.vx v0, v0, t1             // 32-bit vector add for 128 elements
…
```

- Notes:
  - vle8.v: load data into 2 vector registers, even register number or illegal instruction
  - vsext.vf4: the source operands are overlapped with destination operands
    - If source operand is v0, v2, v4, then it is illegal instruction
    - v6 and v7 are legal instructions because the vector registers are read before written by the result data
    - v6 maps to v0-v3, and v7 maps to v4-v7

Copyright© 2020 Andes Technology Corp.

CSR - Vector Byte Length

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7	6 5 4 3 2 1 0
reserved	vlenb = 64

- Vector register length in bytes, vlenb = VLEN/8
  - Design time constant, vlenb = 512/8 = 64
  - Read-only CSR, constant value, hardwired
  - A configuration CSR

Copyright© 2020 Andes Technology Corp.

CSR – Vector Start Index Register (vstart)

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10	9 8 7 6 5 4 3 2 1 0
reserved	vstart[VLMAX*8-1:0]

- Initialize to zero
- vstart is the index of element that causes the trap on vector instruction
- When return from trap, the vector execution restarts from the vstart element and reset the vstart after vector instruction execution
- vstart > vl, meaning that no element operation and vstart is reset to zero

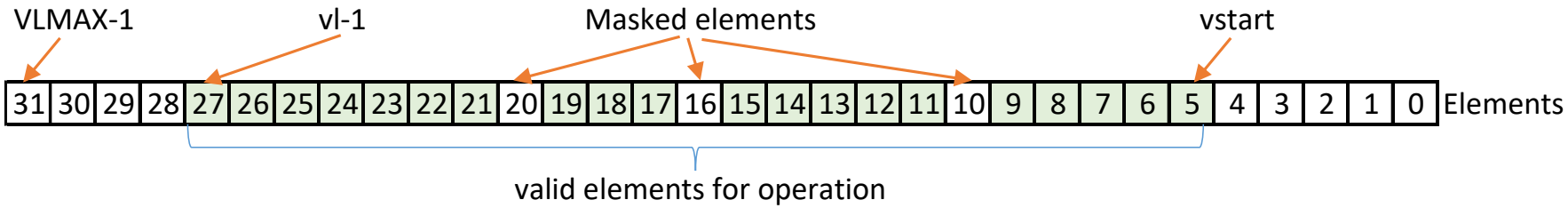
For Andes vector processor, only load/store instruction can cause trap and interrupt are taken at instruction boundary. A non-zero value for vstart for any other vector instruction will cause illegal instruction.

Copyright© 2020 Andes Technology Corp.

Valid Vector Range

Behavior for destination element i

0 < i < vstart	Unchanged
vstart <= i < vl	Updated if v0.mask[i] enabled, unchanged if not mta exception
vl <= i < VLMAX	Unchanged, vta exception



Copyright© 2020 Andes Technology Corp.

Follow Andes, Find Latest Trends

Thank you!

ANDES TECHNOLOGY