Vector Processing

Muhammad Tahir

Lecture 17

Electrical Engineering Department
University of Engineering and Technology Lahore

Contents

Types of Parallelism

- **Instruction-Level Parallelism (ILP)**: Execute multiple independent instructions from one stream (task, thread) in parallel (**pipelining**, **superscalar**, **VLIW**)
- **Data-Level Parallelism (DLP)**: Execute multiple operations of the same type in parallel (**SIMD** (**vector**, **array**) processing)
- **Thread-Level Parallelism (TLP)**: Execute multiple instructions from independent streams in parallel (**multithreading**, **multi-core**)

Flynn's Classification

Flynn's Classification		Data Stream	
		Single	Multiple
Instruction Stream	Single	SISD	SIMD (Vector)
	Multiple	MISD	MIMD (Multi-core)

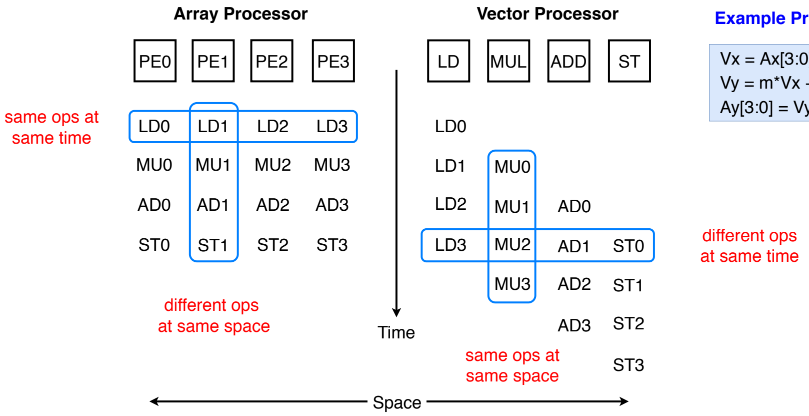
Advantages of SIMD

- Single 32-bit Microprocessor
  - One 32-bit element processing per instruction
- 512-bit SIMD Processor
  - 16 32-bit elements processing per instruction
  - 32 16-bit elements processing per instruction
  - 64 8-bit elements processing per instruction
- For 1 GHz SIMD processor clock frequency the performance is 64 GOPS (for 8-bit elements)

SIMD Processing

- *Single Instruction* operates on *Multiple Data* elements
- Multiple processing elements of same type
- Same instruction operates on multiple data elements at the same time using different spaces (**array processing**)
- Same instruction operates on multiple data elements in consecutive time steps using the same space (**vector processing**)
- *Array-Vector* duality is *Time-Space* duality

SIMD Processing: Vector vs Array

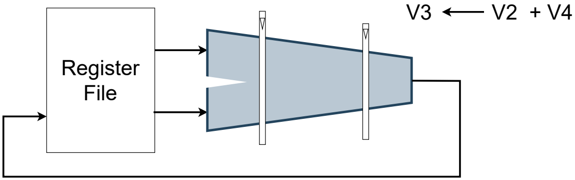


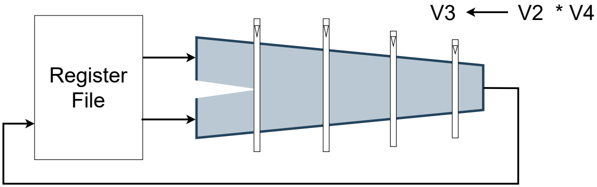
Example Program

Vx = Ax[3:0]
Vy = m*Vx + c
Ay[3:0] = Vy

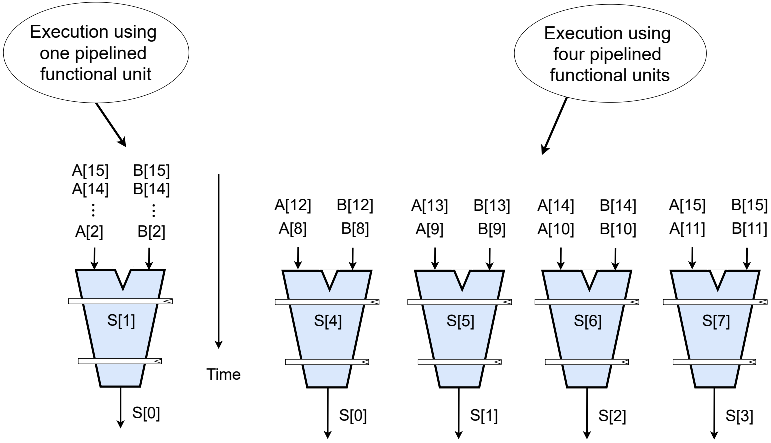
Figure credit: Prof. Onur Mutlu

Vector Arithmetic

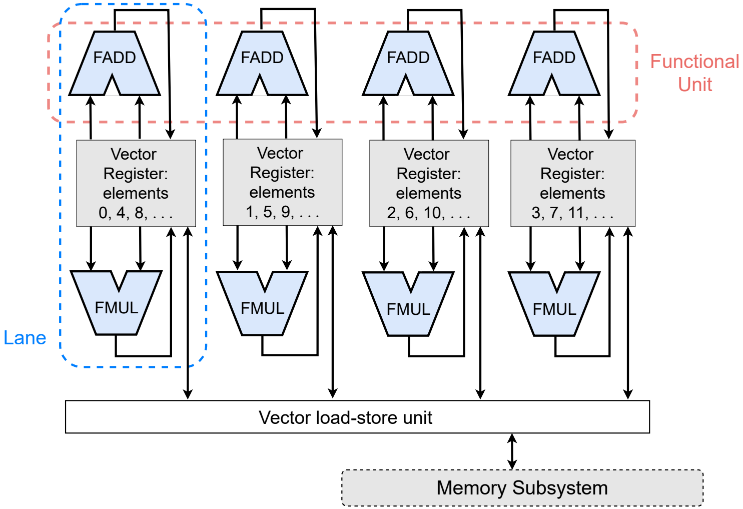




Vector Execution



Structure of Vector Unit

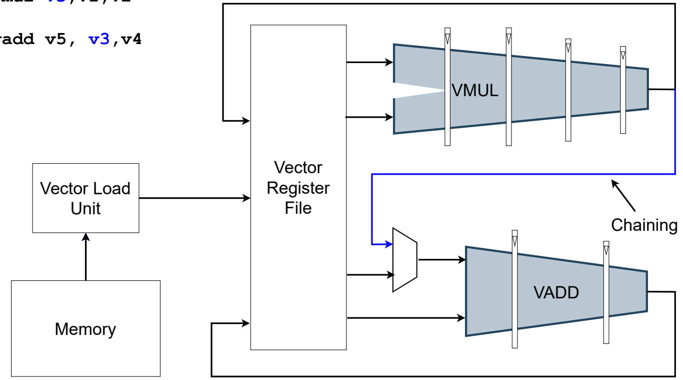

FADD
FADD
FADD
FADD
Functional Unit
Vector Register: elements 0, 4, 8, . . .
Vector Register: elements 1, 5, 9, . . .
Vector Register: elements 2, 6, 10, . . .
Vector Register: elements 3, 7, 11, . . .
FMUL
FMUL
FMUL
FMUL
Lane
Vector load-store unit
Memory Subsystem

Vector Chaining

- Similar to register bypassing

```
vmul v3,v1,v2

vadd v5, v3,v4
```



Vector Stripmining

- Consider an application with data length, N
- Vector registers are of finite length VL_{max} and let $N > \mathrm{VL}_{max}$
- **Stripmining**: Break the loop into segments that fit in vector registers
- For first iteration set

$$\mathrm{VL} = N \bmod \mathrm{VL}_{max}$$

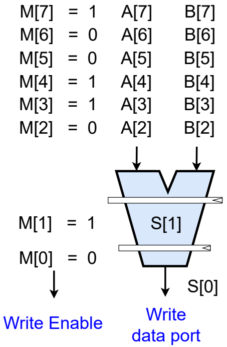
- For second and onward iterations set

$$\mathrm{VL} = \mathrm{VL}_{max}$$

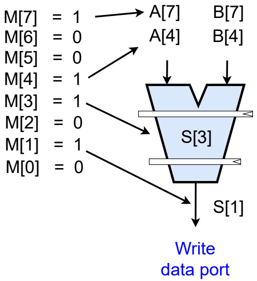
Vector Masking

- Require conditional execution of vector operations
- Introduce **vector mask register**
  - Tells which elements of a vector to be operated on
  - 1-bit masking per element
  - Maskable vector instructions
  - Vector operation becomes *bubble*, when mask bit is clear
- Vector masking implementation
  - **Density Time Implementation**: Scan mask vector and execute nonzero mask elements
  - **Simplified Implementation**: Perform operation on all elements, but control each result writeback based on mask bit

Vector Masking Cont'd



Simplified Implementaton



Density Time Implementaton

RISC V Vector Extension

- RISC V Scalar State
  - Program counter (PC) register
  - 32 XLEN-bit integer registers (x0-x31)
  - 32 FLEN-bit floating point (optional) registers (f0-f31)
  - Control and Status Registers (CSRs) (Privileged Architecture Specs.)
  - Floating point (optional) status register (fCSR) for rounding mode & exception status

RISC V Vector Extension Cont'd

- To support vector extension first define the following three parameters
  - Element Length (**ELEN**): Defines the maximum size of a single vector element in bits, ELEN $\geq$ 8 and must be a power of 2
  - Vector Length (**VLEN**): Defines the number of bits in a vector register, VLEN $\geq$ ELEN and must be a power of 2
  - Stripping Length (**SLEN**): Defines the striping distance in bits, which must comply VLEN $\geq$ SLEN $\geq$ 32 and be a power of 2

RISC V Vector Registers

- Basic Requirements of vector register
  - Each vector data register holds 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 $\geq$ SEW
    - VLMAX – maximum number of vector elements, which can be operated on by single vector instruction (and is equal to LMUL(VLEN/SEW))
  - There are 32 VLEN-bit vector data registers (v0-v31)
  - Vector control registers: VMASK (mask – predication – valid element for execution) is v0

RISC V Vector Registers Cont'd

- RISC V Vector Extension (state augmentation)
  - 32 VLEN-bit vector registers (v0-v31)
  - Vector length register **vl** to specify number of elements to be updated (of destination vector register) by a vector instruction
  - Vector type register **vtype**
  - Vector start register **vstart** for trap handling and specifies the index of first element to be operated by vector instruction
  - Vector byte length register **vlenb** (vector length in bytes)
  - Vector fixed-point rounding mode register **vxrm**
  - Vector fixed-point saturation flag register **vxsat**
  - Vector control and status register **vcsr** (combines **vxrm** and **vxsat** as bit-fields of this register)

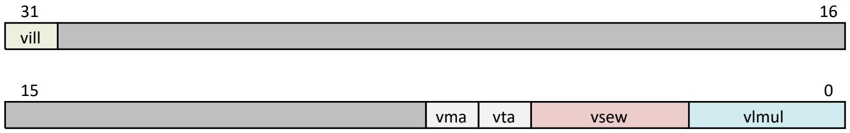
Vector Extension CSRs

Table 1: CSRs for vector extension.

Address	Priv.	Reg. 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)

vtype Register

- **vtype** (vector type) register bit field descriptions



Bit field	**Description**
vlmul[2:0]	Vector register group *length multiplier* (LMUL) setting
vsew[2:0]	Vector *standard element width* (SEW) setting
vta	Vector tail agnostic
vma	Vector mask agnostic
vill	Illegal value if set

vtype Register Cont'd

- The value in **vsew** sets the dynamic *Standard Element Width* (SEW)
- A vector register is viewed as being divided into $\frac{VLEN}{SEW}$ standard-width elements

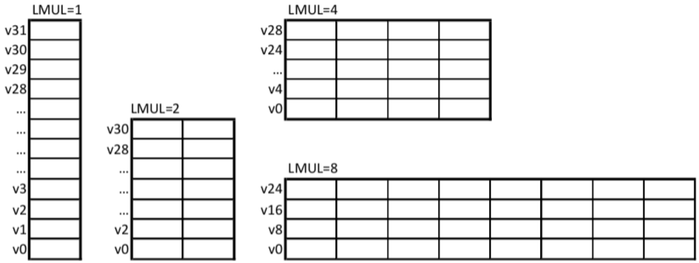
vsew[2:0]	**SEW** $= 8 \times 2^{\mathrm{vsew}}$	**Elements/vector register** (VLEN = 1024 bits)
000	8	128
001	16	64
010	32	32
011	64	16
100	128	8
101	256	4
110	512	2
111	1024	1

vtype Register Cont'd

- The **vlmul** is a signed bit-field
- *Length Multiplier* LMUL $= 2^{\text{vlmul}}$ (for vector register group)
    - Allows multiple vector registers to be grouped together, with single vector instruction operating on multiple vector registers
    - Can be a fractional value, reducing the number of bits used in a vector register

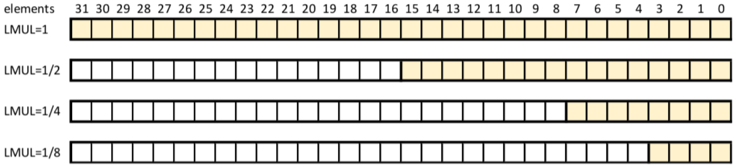
vlmul[2:0]	**LMUL** $= 2^{\text{vlmul}}$	No. of groups	**Indexed register grouping**
100	-	-	Reserved
101 (= -3)	1/8	32	only v_n in the group
110 (= -2)	1/4	32	only v_n in the group
111 (= -1)	1/2	32	only v_n in the group
000	1	32	only v_n in the group
001	2	16	v_n, v_{n+1} in the group
010	4	8	$v_n, \cdots, v_{n+3}$ in the group
011	8	4	$v_n, \cdots, v_{n+7}$ in the group

vtype Register Cont'd



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

vtype Register Cont'd

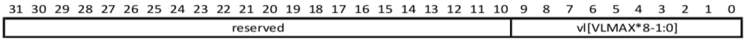


- LMUL=1/2, load data into half vector register, then perform vector double operations to expand data to the whole register
- LMUL=1/4, using vector extension of 4x to expand data to the whole register
- LMUL=1/8, using vector extension of 8x to expand data to the whole register

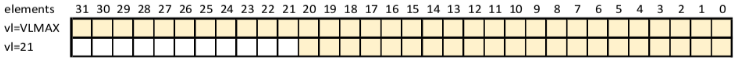
vtype Register Cont'd

- **vta** (*vector tail agnostic*) bit controls updating of destination tail elements by a vector instruction
- **vma** (*vector mask agnostic*) bit allows masking of destination elements by a vector instruction
- The **vill** bit tells that a vector instruction attempted to write an unsupported value to **vtype**
- If **vill** bit is set, then any attempt to execute a vector instruction that depends upon **vtype** will raise an illegal-instruction exception
- When the **vill** bit is set, the other XLEN-1 bits in **vtype** should be zero

Vector Length Register **vl**



- vl is the number of elements for vector operations
- vl is set to VLMAX if the set value is greater than VLMAX
- The default vl is VLMAX
- vl = 0, then no elements are updated in vector operations



VMASK Register

- 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
  - 1 – unmask
- The mask is enabled for each vector instruction.
  - vop.v v8, v16, 3, v0.t // mask is enabled, each element writes back with v0.mask[i] = 1
  - vop.v v8, v16, 3 // unmask, all elements write result data back to VRF

RV V extension ISA

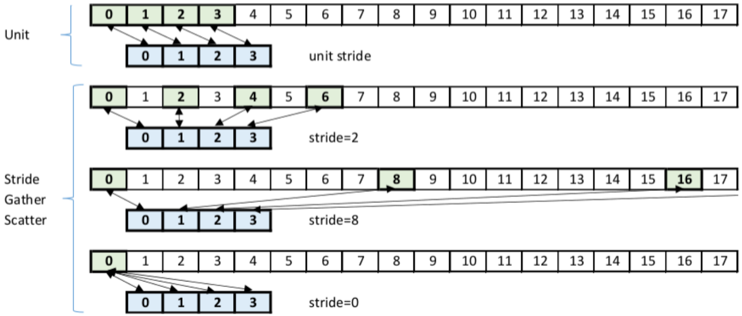
- RV V extension ISA includes:
    - Memory Operations
    - Compute Instructions
    - Permute Instructions

RVV ISA Memory Operations

- Load/store operations move groups of data between the VRF and memory
- Four basic types of addressing is used in RVV
  - Unit stride: fetch consecutive elements in contiguous memory
  - Non-unit or constant stride: elements are in stride memory; stride is the distance between 2 elements of a vector in memory
  - Indexed (gather-scatter): elements are randomly offset from a base address
  - Segment: move multiple contiguous fields in memory to and from consecutively numbered vector registers
- Note that Atomic load/store is not a part of vector ISA

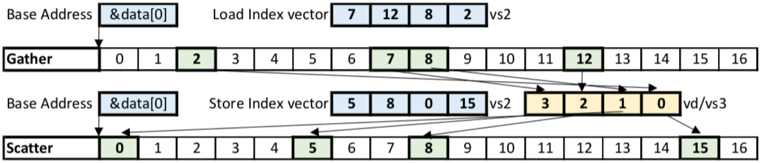
RVV ISA Memory Operations

- Unit stride continuous memory access
- Constant stride: gather scatter (stride can be negative or zero)



RVV ISA Memory Operations

- The indices are absolute and not accumulative
- The indices are not in-order



Vector Load/Store Examples

- SEW=16, LMUL=1, VLEN=512b
- elements=512/16=32, all vector instructions operate on 32 elements
    - vle16.v v1, (t0) // Load a vector of sew = 16, lmul = 1, load to v1
    - vle32.v v2, (t1) // Load a vector of sew = 32, lmul = 2, load to v2 and v3
    - vse32.v v4, (t1) // Store a vector of sew = 32, emul = 2, v4 and v5 store to memory
    - vle8.v v1, (t1) // Load a vector of sew=8, emul = 1/2, load to elements first half of v1
    - vle64.v v4, (t0) // Load a vector of sew=64, emul = 4, load to v4, v5, v6, v7

RVV Compute Instructions

- RVV compute instructions includes:
    - Vector Arithmetic Instructions
    - Vector Widening Instructions
    - Vector Narrow Instructions
    - Vector-integar Arithmetic Instructions
    - Vector Bitwise Logical Instructions
    - Vector Bit Shift Instructions
    - Vector Integer Comparison Instructions
    - Vector Integer Min/Max Instructions
    - Vector Mask-Register Logical Instructions

Suggested Reading

- RISC-V Vector Extension Webinar II [RISC-V Vector Extension Webinar II]
- Lecture slides on Vector extension [Asanovic, 2020a]
- Lecture slides on RISC V Vector extension [Asanovic, 2020b]
- RISC-V "V" Vector Extension Specifications, Draft Version 0.9 [Asanovic et al., 2016]
- RISC-V Vector Extension Webinar III [RISC-V Vector Extension Webinar III]

References

Asanovic, K. (2020a).
"risc-v vectors, cs152, spring 2020".
http://inst.eecs.berkeley.edu/~cs152/sp20/lectures/L15-Vectors.pdf.

Asanovic, K. (2020b).
"risc-v vectors, cs152, spring 2020".
http://inst.eecs.berkeley.edu/~cs152/sp20/lectures/L17-RISCV-Vectors.pdf.

Asanovic, K., Waterman, A., and et. al. (2016).
The risc-v "v" vector extension specifications, draft version 0.9.

RISC-V Vector Extension Webinar II.
"RISC-V Vector Extension Webinar II, Andes Technology".
http://www.andestech.com/wp-content/uploads/Andes-RVV-Webinar-II_final.pdf.

RISC-V Vector Extension Webinar III.
"RISC-V Vector Extension Webinar III, Andes Technology".
http://www.andestech.com/wp-content/uploads/