CS 152 Computer Architecture and Engineering
CS252 Graduate Computer Architecture

Lecture 16 – RISC-V Vectors

Krste Asanovic
Electrical Engineering and Computer Sciences
University of California at Berkeley

http://www.eecs.berkeley.edu/~krste
http://inst.eecs.berkeley.edu/~cs152

Last Time in Lecture 15

Vector supercomputers

- Vector register versus vector memory
- Scaling performance with lanes
- Stripmining
- Chaining
- Masking
- Scatter/Gather

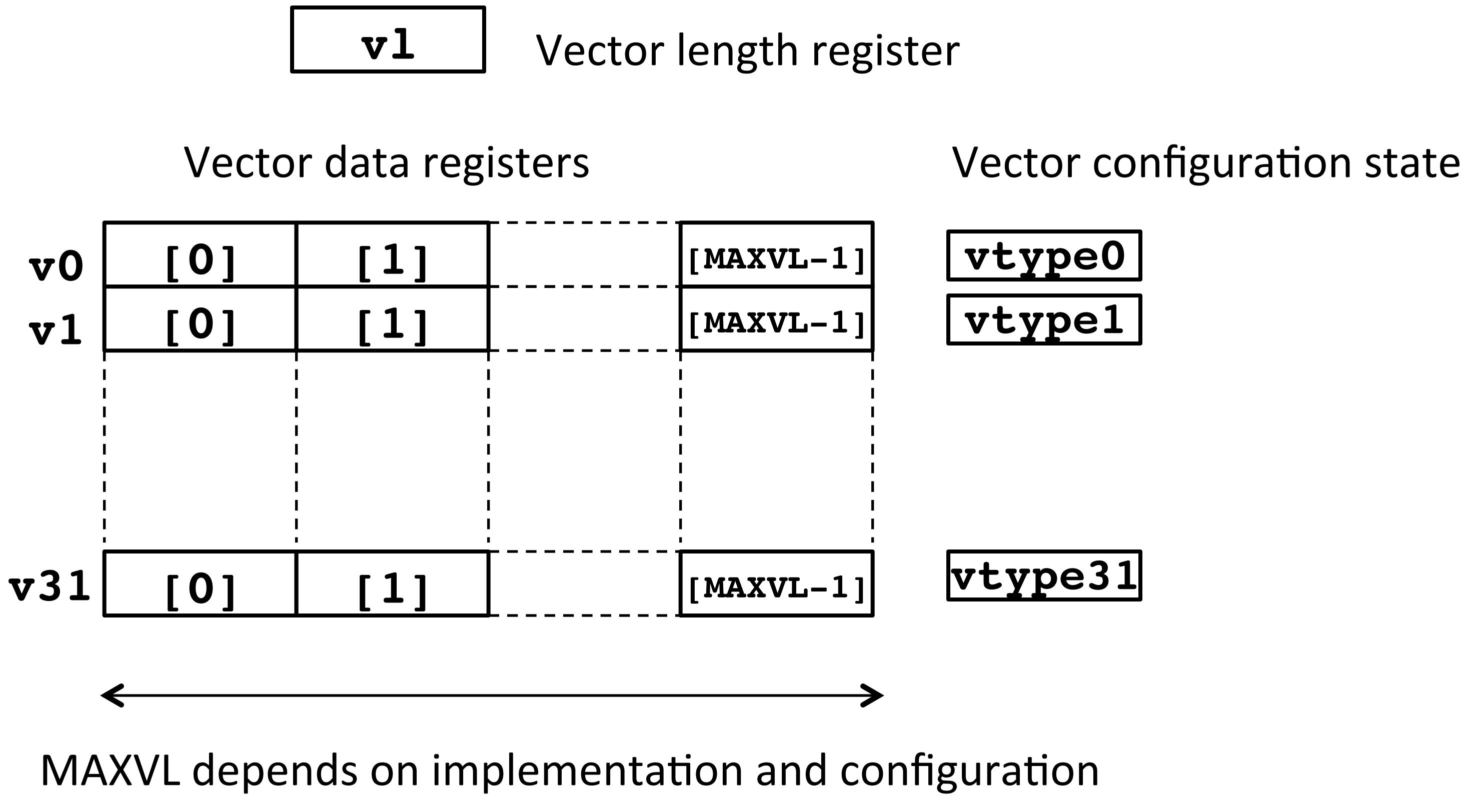
New RISC-V Vector Standard

- Being added as a standard extension to the RISC-V ISA
- An updated form of Cray-style vectors for modern microprocessors
- Still a work in progress, so details might change before being standardized
- Today, a short tutorial on vector programming using the current draft standard
- Assembly syntax will be changed, but concepts should be correct

Quick Summary of RISC-V Vector ISA

- 32 vector registers, v0-v31 (no zero register)
- Each vector register has associated type including:
  - Number of bits in each element (8-bit, 16-bit, 32-bit, 64-bit, 128-bit, …)
  - Representation (signed integer, unsigned integer, floating-point)
  - Shape (scalar, 1D vector)
- Number of vector registers, and their type are configured with special instructions before using vector unit
- Vector instructions are *polymorphic*, operation depends on opcode and vector register operand types
- Vector length register, vl , controls number of elements executed by each instruction
- Instructions can be executed under mask
- Vector arithmetic operations include all standard scalar operations
- Vector memory operations include unit-stride, constant-stride, scatter-gather

Vector Unit State



Vector Configuration

- Can be done by writing a number of control registers
- Faster method is a special instruction that sets up standard patterns:
  - vcfg N1*TYPE1, N2*TYPE2, N3*TYPE3,…
  - v0..vN1-1 have TYPE1, vN1…v(N1+N2-1) have TYPE 2, …

- Requested vector registers are zeroed
- Unused vector registers are not accessible
- MAXVL depends on number of vector registers and type, but can write assembly code without knowing MAXVL

`setvl` instruction

```
setvl xregd, xregsrc
```

- **`vl`** is set to MIN(MAXVL, xregsrc), also copied to xregd
- E.g., **`setvl t0,a0 # vl, t0 = MIN(MAXVL,a0)`**

Simple Example: Vector-vector add

```
for (i=0; i<N; i++)
{ C[i] = A[i] + B[i]; } // 32-bit ints

  vcfg 3*V32bINT # Set up v0,v1,v2 to be 32-bit ints
Loop:
  setvl t0, a0 # a0 holds N, t0 holds amount done
  ld v0, a1    # load strip of A vector
  ld v1, a2    # load strip of B vector
  vadd v2, v0, v1 # add vectors
  st v2, a3    # store strip of C vector
  slli t1, t0, 2 # multiply by 4 to get bytes
  add a1, a1, t1 # bump pointers
  add a2, a2, t1
  add a3, a3, t1
  sub a0, a0, t0 # Subtract amount done
  bnez a0, Loop
```

Simple Example: Float vector-vector add

```
for (i=0; i<N; i++)
{ C[i] = A[i] + B[i]; } // 32-bit ints

  vcfg 3*V32bFP # Set up v0,v1,v2 to be 32-bit floats
Loop:
  setvl t0, a0 # a0 holds N, t0 holds amount done
  ld v0, a1    # load strip of A vector
  ld v1, a2    # load strip of B vector
  vadd v2, v0, v1 # add vectors
  st v2, a3    # store strip of C vector
  slli t1, t0, 2 # multiply by 4 to get bytes
  add a1, a1, t1 # bump pointers
  add a2, a2, t1
  add a3, a3, t1
  sub a0, a0, t0 # Subtract amount done
  bnez a0, Loop
```

Vector-vector add, 32b+16b -> 32b integer

```
for (i=0; i<N; i++)
{ C[i] = A[i] + B[i]; } // 32-bit ints

  vcfg 2*V32bINT, 1*V16bINT # v0,v1:32b, v2:16b
Loop:
  setvl t0, a0 # a0 holds N, t0 holds amount done
  ld v0, a1     # load strip of A vector
  ld v2, a2     # load strip of B vector
  vadd v1, v0, v2 # add vectors
  st v1, a3     # store strip of C vector
  slli t1, t0, 2 # multiply by 4 to get bytes
  slli t2, t0, 1 # multiply by 2 to get bytes
  add a1, a1, t1 # bump pointers
  add a2, a2, t2
  add a3, a3, t1
  sub a0, a0, t0 # Subtract amount done
  bnez a0, Loop
```

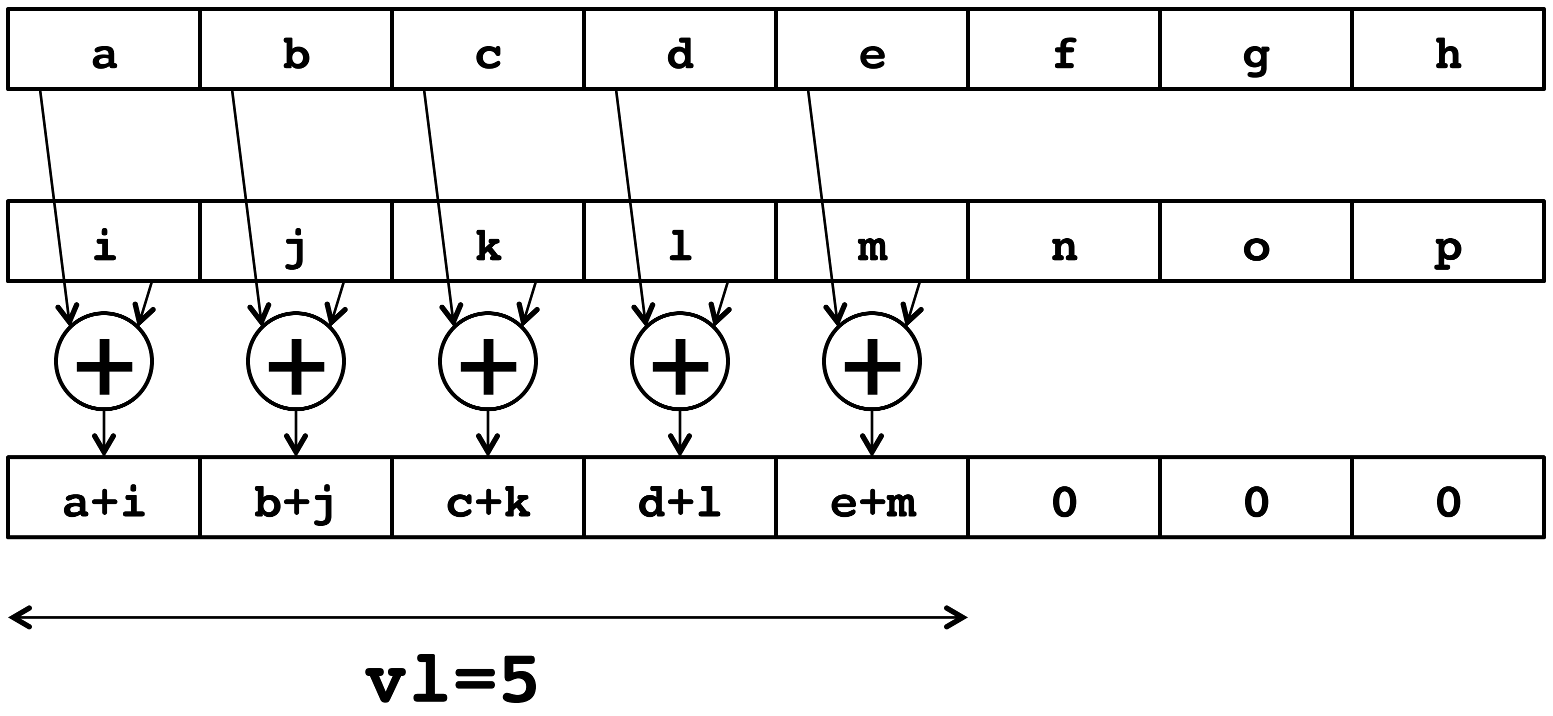
Vector-scalar add

```
for (i=0; i<N; i++)
{ C[i] = A[i] + B; } // 32-bit ints

  vcfg 2*V32bINT, 1*S32bINT #
  vmv v2, a2    # Copy B to vector unit scalar
Loop:
  setvl t0, a0 # a0 holds N, t0 holds amount done
  ld v0, a1    # load strip of A vector
  vadd v1, v0, v2 # add vectors
  st v1, a3    # store strip of C vector
  slli t1, t0, 2 # multiply by 4 to get bytes
  add a1, a1, t1 # bump pointers
  add a3, a3, t1
  sub a0, a0, t0 # Subtract amount done
  bnez a0, Loop
```

Vector length < MAXVL

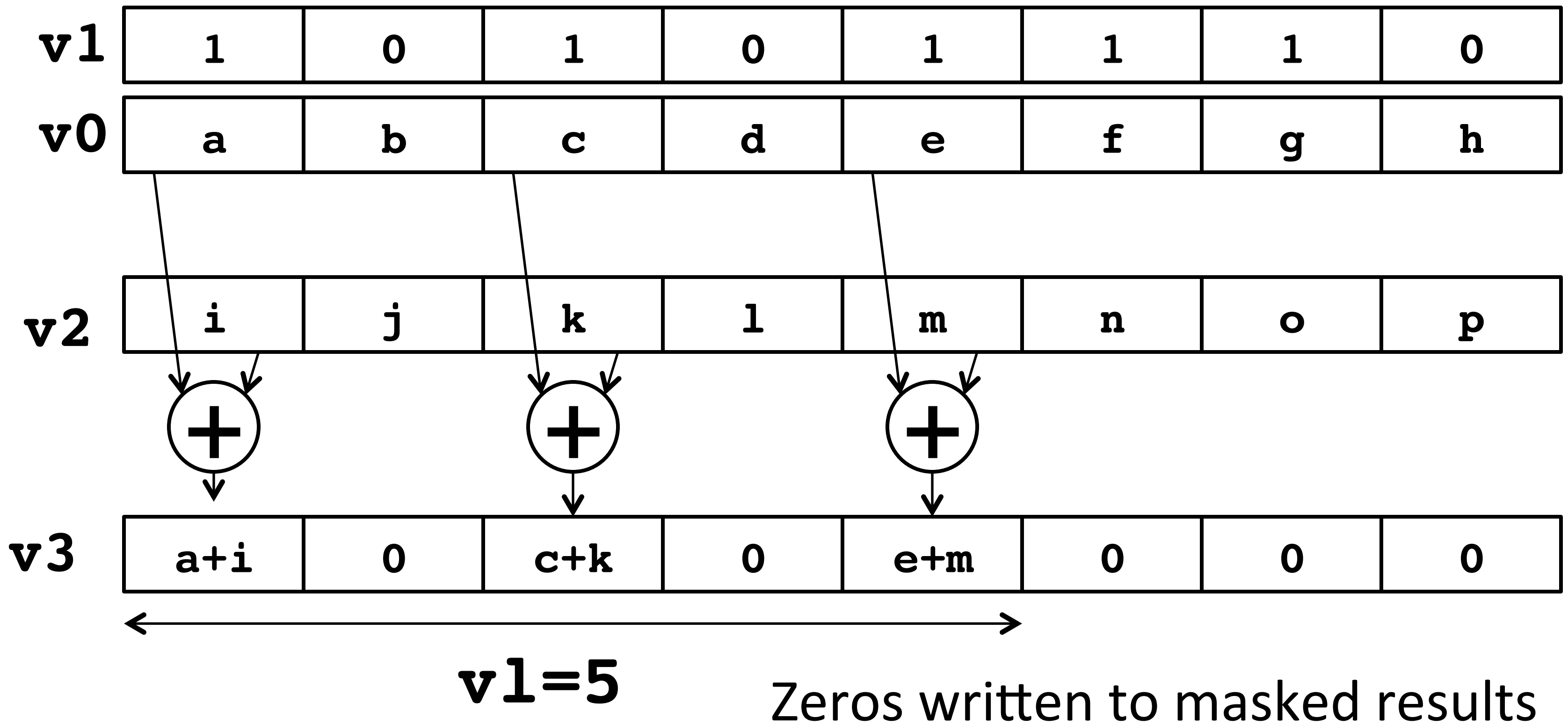
- When vl is set to less than MAXVL, elements past end of vector are zeroed. This supports vector register renaming.
- E.g., MAXVL=8, vl = 5, when execute add:



Vector masking

- In base instruction set, instructions can be "unmasked" or masked by vector register v1, either true or complement
- Least-significant bit of each element is used as mask

```
vadd v3,v0,v2,v1.t # Execute only if v1.LSB is set
```



Zeros written to masked results

CS152 Administrivia

- PS 4 due Friday March 23 in section
  - Can also turn in on class Wednesday, office hours, or can email pdf

CS252 Administrivia