ELE 655
Microprocessor System Design

Section 3 – Data Level Parallelism

Class 2 – SIMD

DLP
Data Level Parallelism

- SIMD overview
  - Why SIMD
  - Many types of calculations require less than the full register size of a 32bit or 64 bit processor
    - Video – typically 8 bits for color, …
    - Audio – 12-16 bits
  - Vector processors are relatively big and complex
  - Can we leverage the existing processor functionality?

DLP
Data Level Parallelism

- SIMD overview
  - Current generations are added as a co-processor
  - Typically implemented as extensions to an instruction set
    - Load, Store, Arithmetic
    - No inherent control capabilities
  - Often incorporated with the floating point hardware
  - Similar to VLIW but:
    - Fixed operand lengths (no Vector length register)
    - Limited addressing modes (no stride or gather/scatter)
    - No masking capability

DLP
Data Level Parallelism

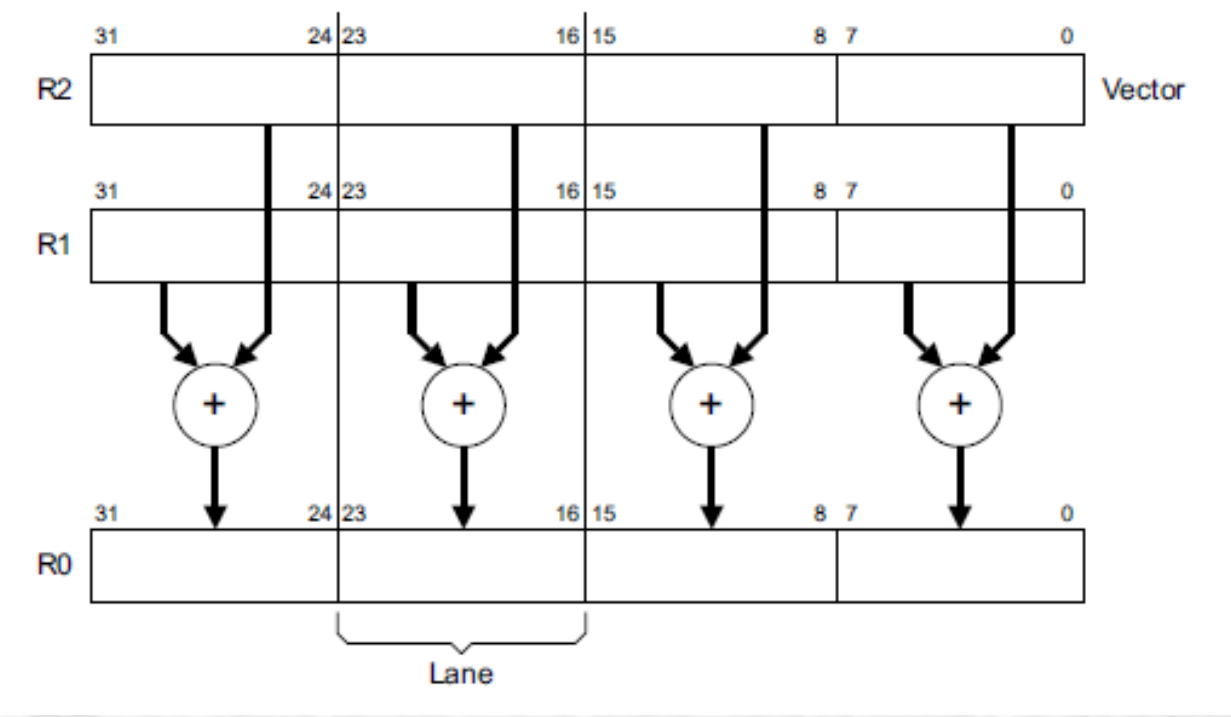
- SIMD overview
  - Limitations in the instruction set lead to
    - Poor compile target
    - Dominated by assembly code
  - Large sets of libraries developed for specific functions

DLP
Data Level Parallelism

- SIMD overview
  - Basic Idea
    - Take a large register (32-64 + bits)
      - Break it into a set of smaller registers
      - 64, 2x32, 4x16, 8x8
    - Modify the arithmetic units into matching sub-blocks
      - 64 bit adder → 8 - 8 bit adders
      - Allow the HW to operate individually or in groups (with no penalty)
      - 8 – 8 bit adds, 4 – 16 bit adds, 2 – 32 bit adds, 1 – 64 bit add
    - Add separate instructions (or operands) for each case

DLP
Data Level Parallelism

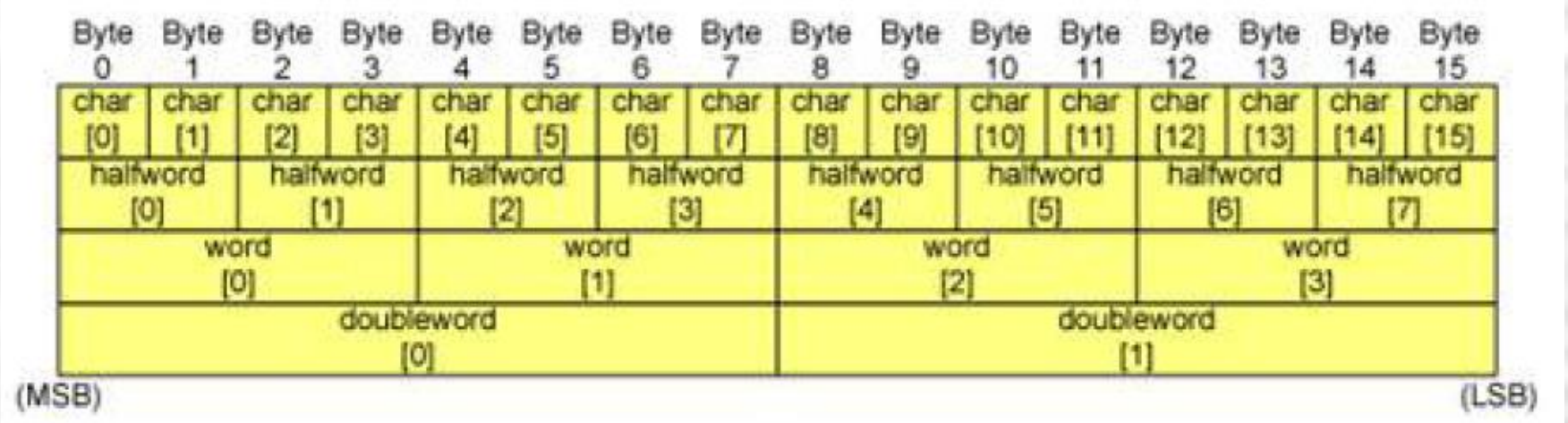
- SIMD overview
  - Basic Idea
    - 32 bit registers and ALU



DLP

Data Level Parallelism

- SIMD
  - 128 Bit SIMD register



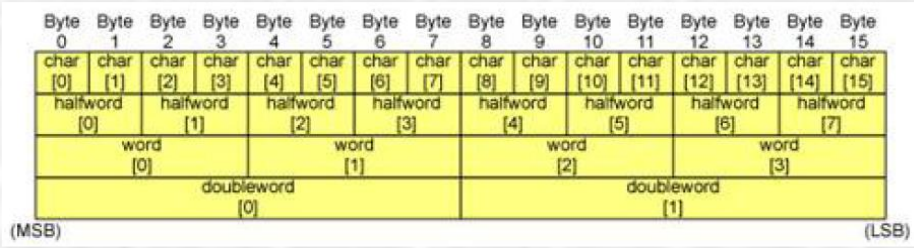
DLP
Data Level Parallelism

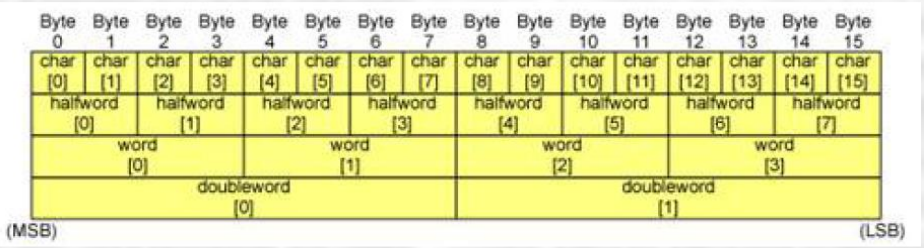
- SIMD
  - Intel
    - MMX (1996)
      - 64 bit floating point registers
      - X8, x4, x1 configurations
    - SSE (1999)
      - 128 bit SIMD specific registers
      - X16, x8, x4, x2 configurations
    - AVX (2010)
      - 256 bit SIMD specific registers
      - X32, x16, x8, x4
      - Prepared to extend to 512 bits and 1024 bits

DLP
Data Level Parallelism

- SIMD
  - AVX SIMD Register
    - 256 bits

Category	Operands
Unsigned Add/Subtract	32-8bit, 16-16bit, 8-32bit, 4-64bit
Floating Point	16-16bit, 8-32bit, 4-64bit, 2-128bit





DLP
Data Level Parallelism

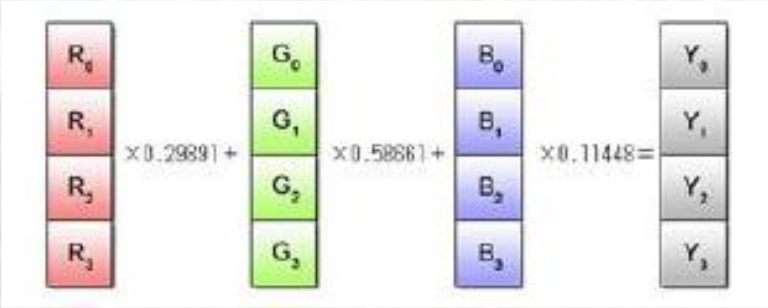
- SIMD
  - MIPS Example
    - 256 bits

```
        L.D       F0,a          ;load scalar a
        MOV       F1, F0        ;copy a into F1 for SIMD MUL
        MOV       F2, F0        ;copy a into F2 for SIMD MUL
        MOV       F3, F0        ;copy a into F3 for SIMD MUL
        DADDIU    R4,Rx,#512    ;last address to load
Loop:   L.4D      F4,0[Rx]      ;load X[i], X[i+1], X[i+2], X[i+3]
        MUL.4D    F4,F4,F0      ;a×X[i],a×X[i+1],a×X[i+2],a×X[i+3]
        L.4D      F8,0[Ry]      ;load Y[i], Y[i+1], Y[i+2], Y[i+3]
        ADD.4D    F8,F8,F4      ;a×X[i]+Y[i], ..., a×X[i+3]+Y[i+3]
        S.4D      0[Ry],F8      ;store into Y[i], Y[i+1], Y[i+2], Y[i+3]
        DADDIU    Rx,Rx,#32     ;increment index to X
        DADDIU    Ry,Ry,#32     ;increment index to Y
        DSUBU     R20,R4,Rx     ;compute bound
        BNEZ R20,Loop           ;check if done
```

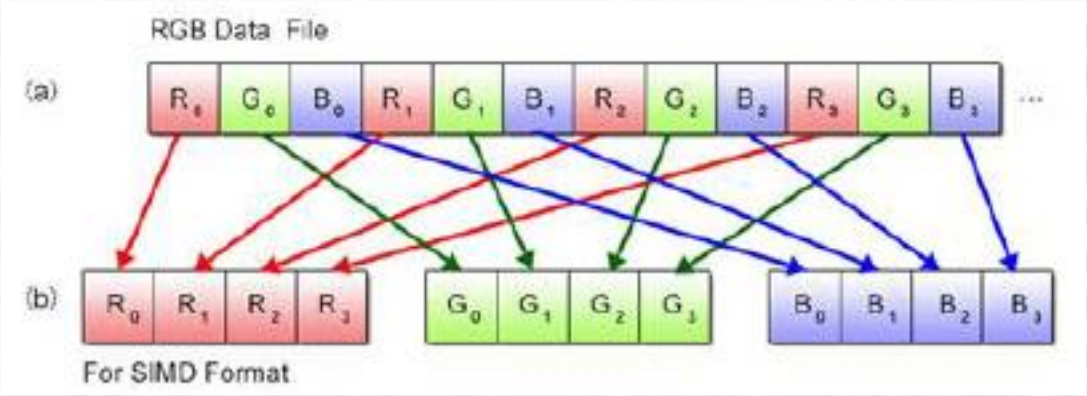
~ 4x reduction in instructions over non-SIMD

DLP
Data Level Parallelism

- SIMD
  - RGB



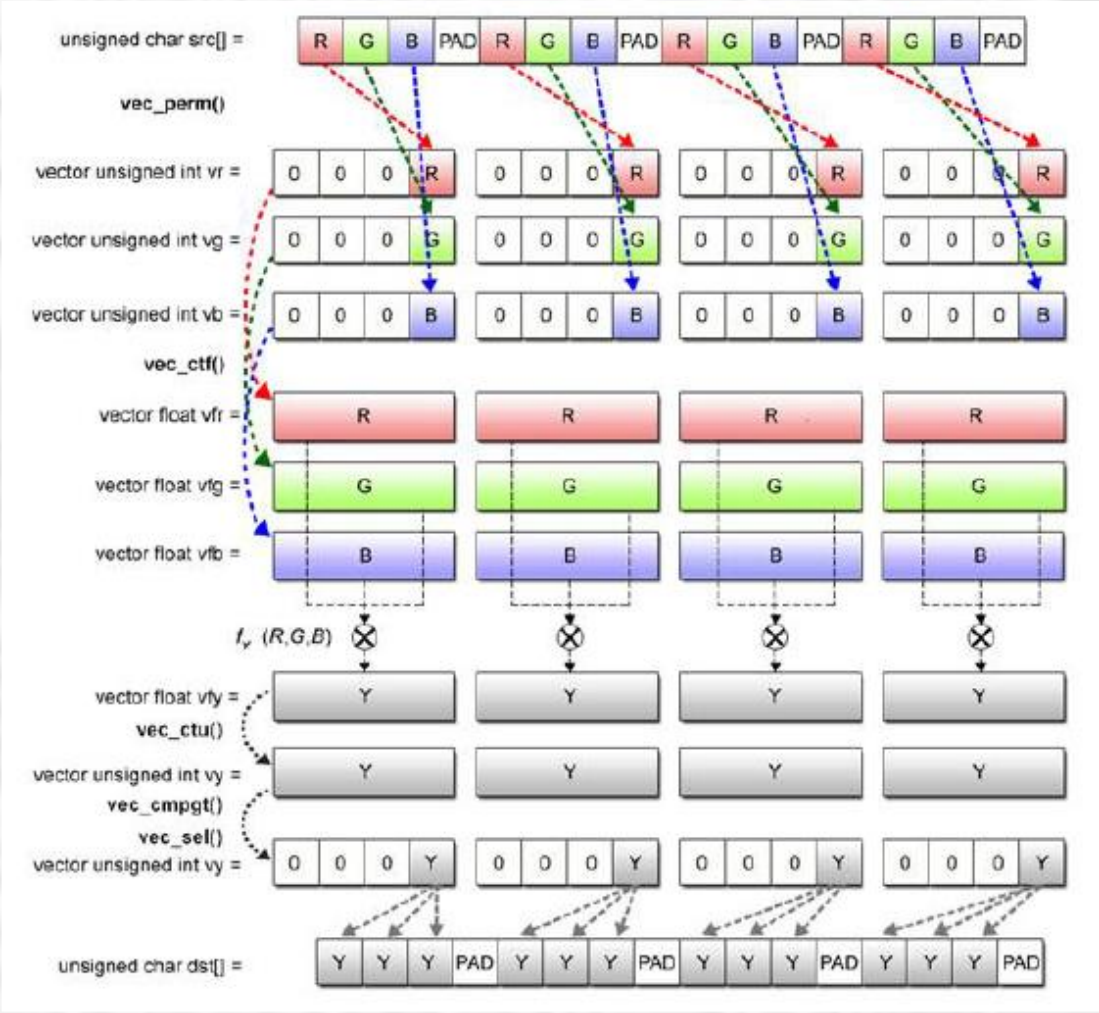
  - Vector realignment



DLP

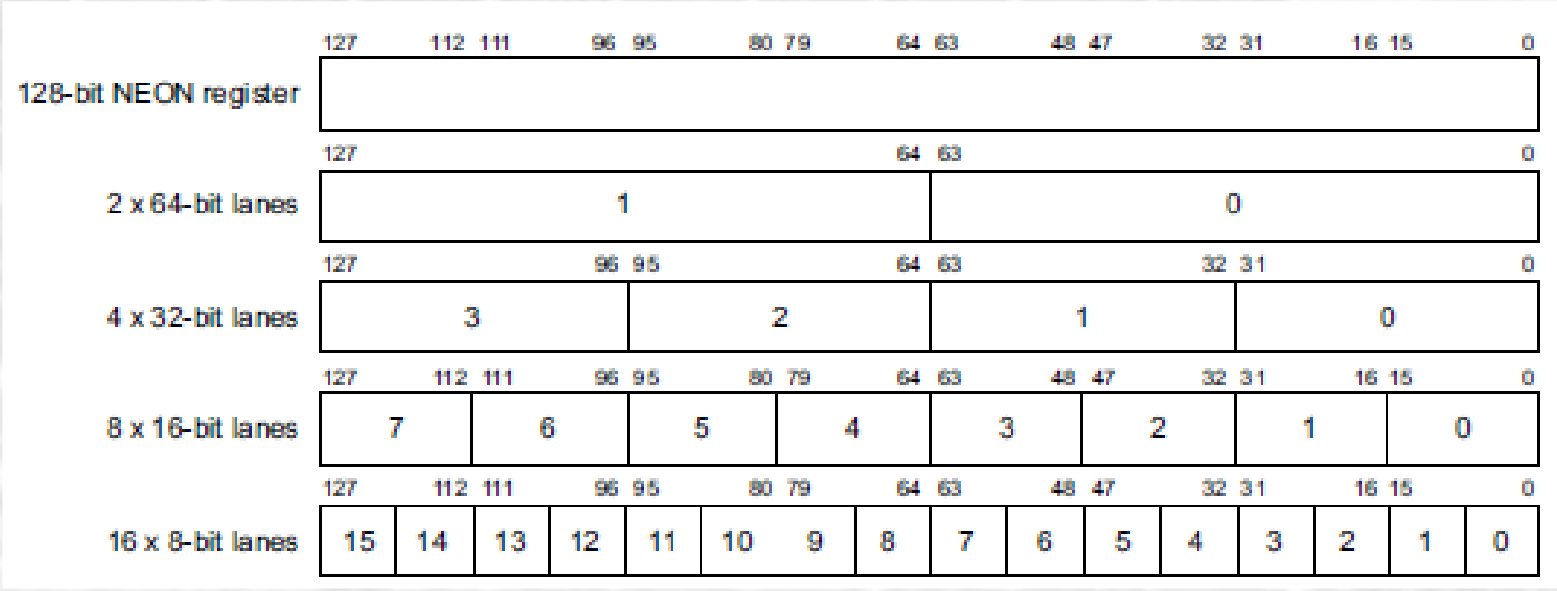
Data Level Parallelism

- SIMD
  - RGB Brightness



DLP
Data Level Parallelism

- SIMD
  - ARM - NEON
    - 32 – 128 bit Vector registers
    - ≤ 16 lanes – depending on instruction



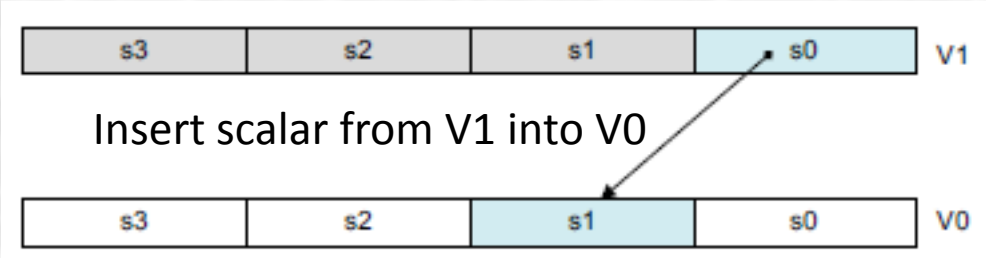
DLP
Data Level Parallelism

- SIMD
  - ARM - NEON
    - Scalar access within the vector
    - The general array notation to access individual elements of a vector is:
      - <Instruction> Vd.Ts[index1], Vn.Ts[index2]
      - where:
      - *Vd* is the destination register.
      - *Vn* is the first source register.
      - *Ts* is the size specifier for the element.
      - *index* is the element index.

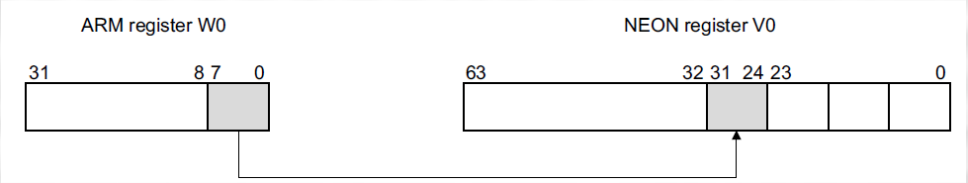
DLP
Data Level Parallelism

- SIMD
  - ARM - NEON
    - Scalar access within the vector

INS V0.S[1], V1.S[0]



MOV V0.B[3], W0



DLP
Data Level Parallelism

- SIMD
  - ARM - NEON
    - Instruction Format

{prefix}op{suffix} Rd.T Rs.T {Rs.T}

    - Prefix
      - Signed (s)
      - Unsigned (u)
      - Floating Point (f)
      - Polynomial (p)

DLP
Data Level Parallelism

- SIMD
  - ARM - NEON
    - Instruction Format

{prefix}op{suffix} Rd.T Rs.T {Rs.T}

    - T
      - 8 – bytes (8B)
      - 16 – bytes (16B)
      - 4 – half words (4H)
      - 8 – half words (8H)
      - Words (S)
      - 2 – words (2S)
      - 4 – words (4S)
      - 2 – double words (2D)

DLP
Data Level Parallelism

- SIMD
  - ARM - NEON
    - Instruction Format

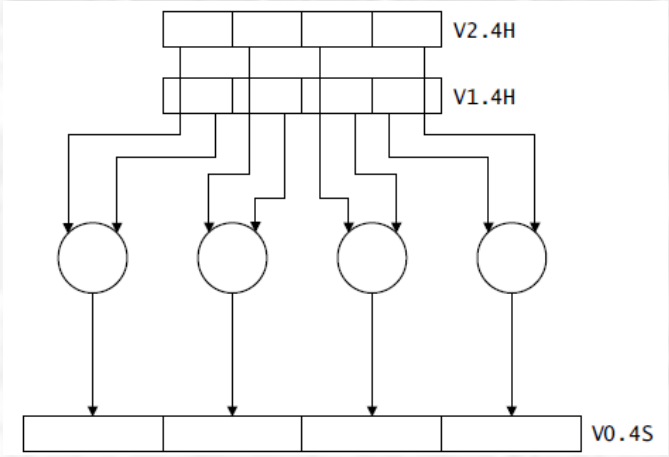
{prefix}op{suffix} Rd.T Rs.T {Rs.T}

    - suffix
      - Normal ()
      - Long (l)
      - Wide (w)
      - Narrow (n)
      - Pairwise (p)
      - Across lanes (v)
      - Upper Half (2)

DLP
Data Level Parallelism

- SIMD
  - ARM – NEON - suffix
    - Normal – maintains the width of the vector
    - Long – result is twice as long as operands

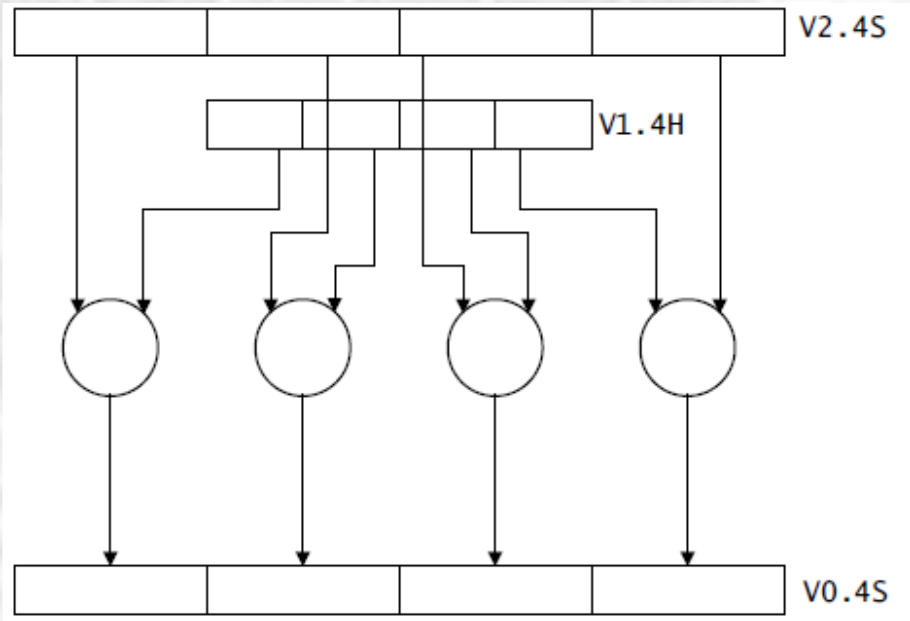
SADDL V0.4S, V1.4H, V2.4H



DLP
Data Level Parallelism

- SIMD
  - ARM – NEON - suffix
    - Wide – Single word operand and double word operand → double word result

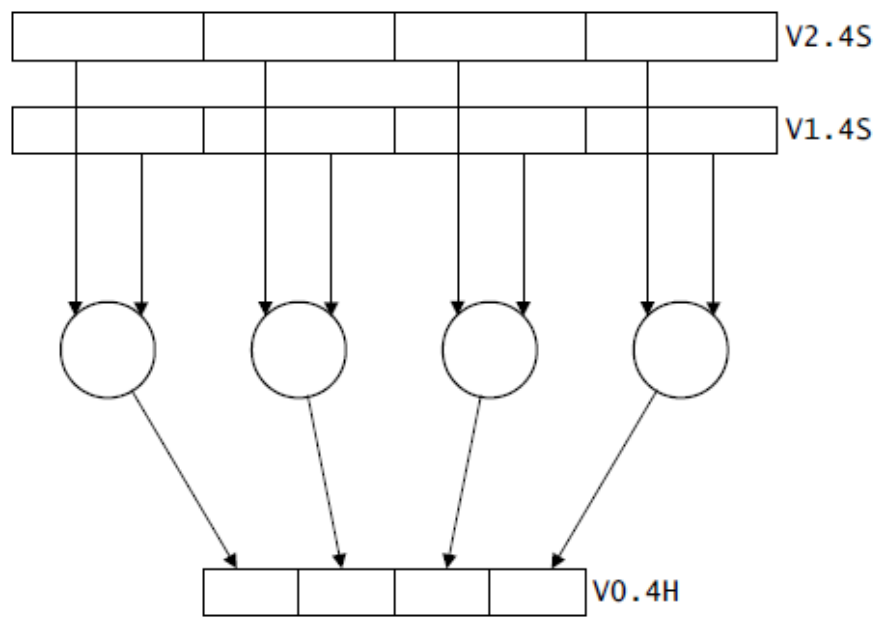
SADDW V0.4S, V1.4H, V2.4S



DLP
Data Level Parallelism

- SIMD
  - ARM – NEON - suffix
    - Narrow – Single word result from double word operands

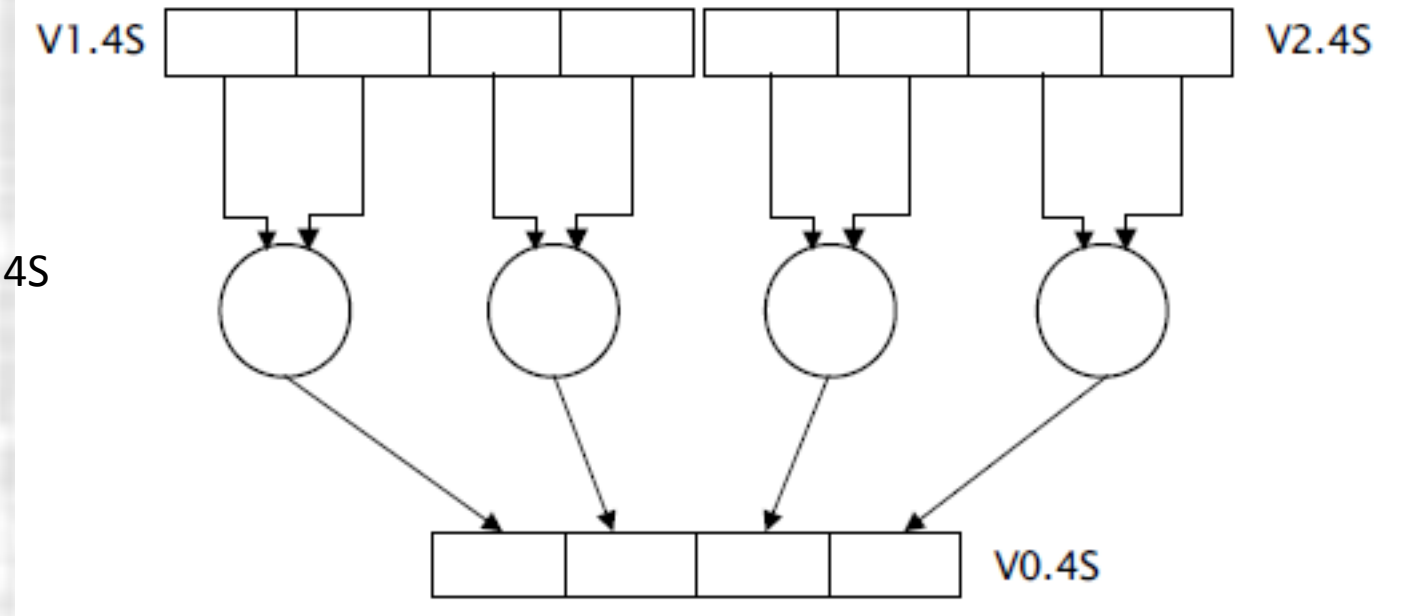
SUBHN V0.4H, V1.4S, V2.4S



DLP
Data Level Parallelism

- SIMD
  - ARM – NEON - suffix
    - Pairwise – operate on pairs of elements

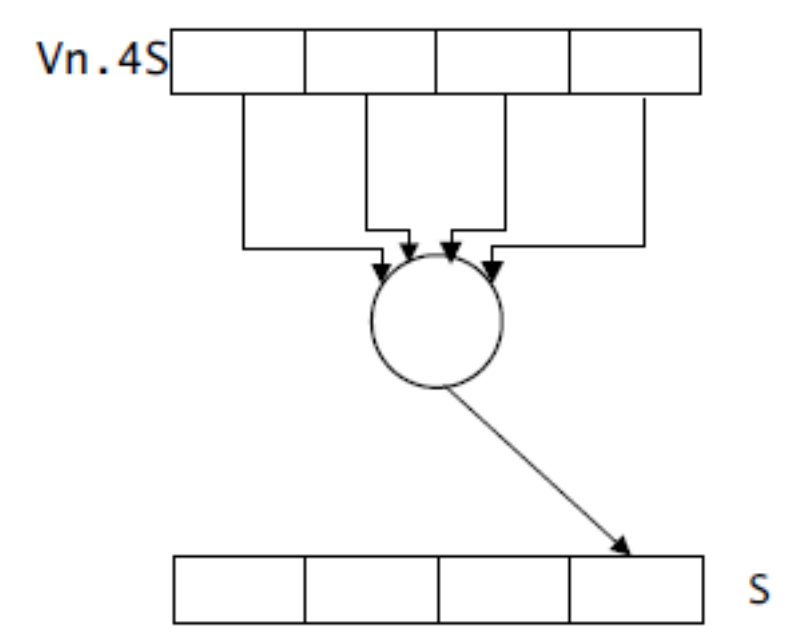
ADDP V0.4S, V1.4S, V2.4S



DLP
Data Level Parallelism

- SIMD
  - ARM – NEON - suffix
    - Across all lanes – operate across all elements

ADDV S0, V1.4S



DLP
Data Level Parallelism

- SIMD
  - ARM – NEON - suffix
    - Upper half – operate on the high order elements

SADDW2 V0.2D, V1.2D, V2.4S