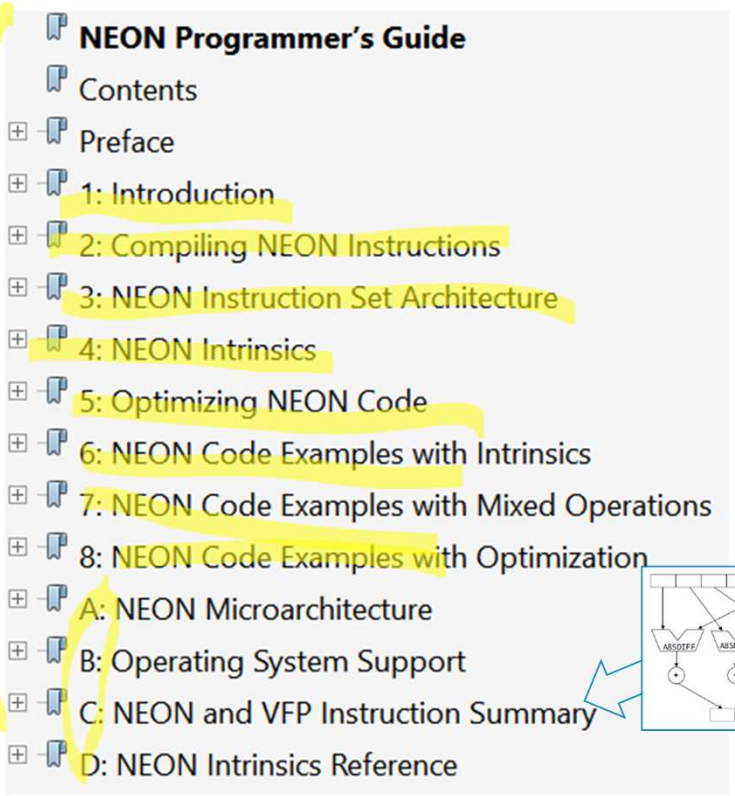
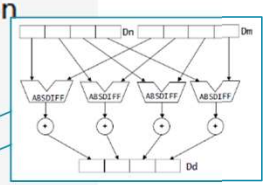
NC STATE UNIVERSITY

Introducing NEON Advanced SIMD Processing

References

- NEON Programmer's Guide DEN0018 (NPG) – read this first!





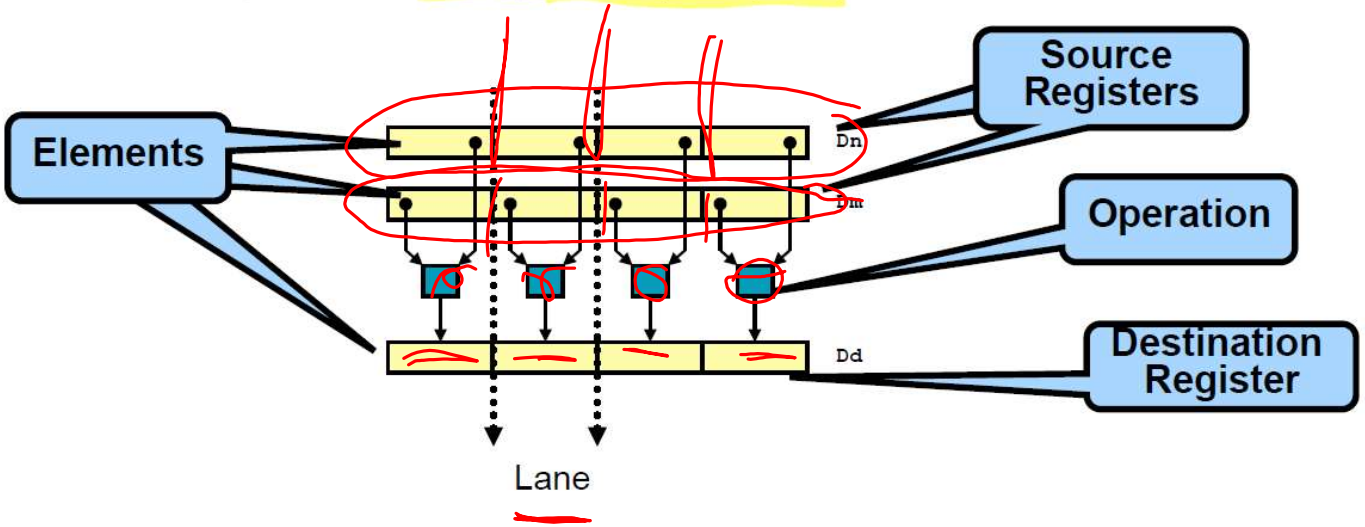
- Instr. Functionality: ARM Arch. Ref. Manual
  - Load/Store: 4.11
  - Register Transfer: 4.12
  - Data Processing: 4.13, 4.14
- ARM C Language Extensions IHI0053 (ACLE)
- ARM NEON Intrinsics Reference IHI0073 (NIR)
- Performance: Cortex-A72 Software Optimization Guide UAN0016

What is NEON?

NEON technology is the implementation of the ARM Advanced *Single Instruction Multiple Data* (SIMD) extension.

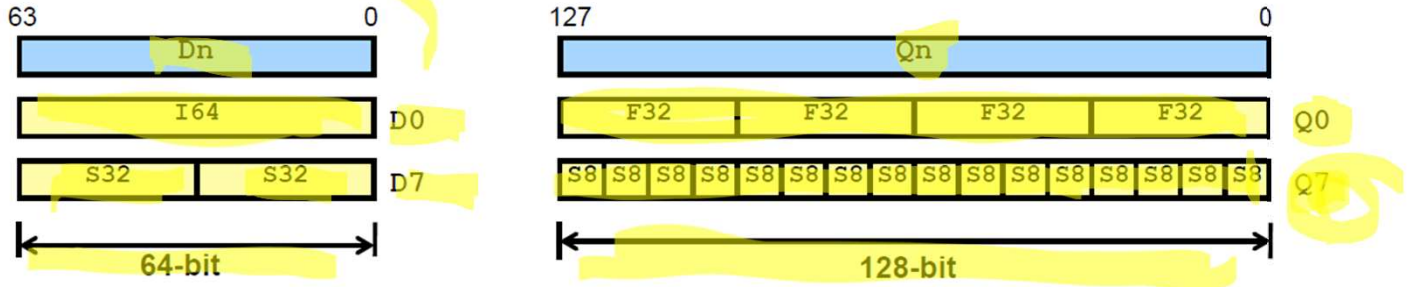
The NEON unit is the component of the processor that executes SIMD instructions. It is also called the NEON *Media Processing Engine* (MPE).

- NEON is a wide SIMD data processing architecture
  - Extension of the ARM instruction set
  - 32 registers, 64-bits wide (dual view as 16 registers, 128-bits wide)
- NEON Instructions perform "Packed SIMD" processing
  - Registers are considered as vectors of elements of the same data type
  - Data types can be: signed/unsigned 8-bit, 16-bit, 32-bit, 64-bit, single prec. float
  - Instructions perform the same operation in all lanes

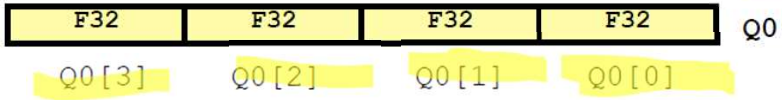


Vectors and Scalars

- Registers hold one or more elements of the same data type
  - Vn can be used to reference either a 64-bit Dn or 128-bit Qn register
  - A register, data type combination describes a vector of elements



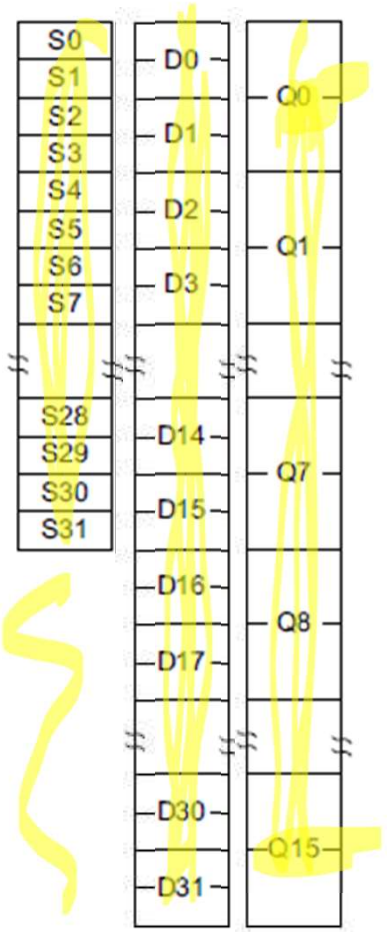
- Some instructions can reference individual scalar elements
  - Scalar elements are referenced using the array notation `Vn[x]`



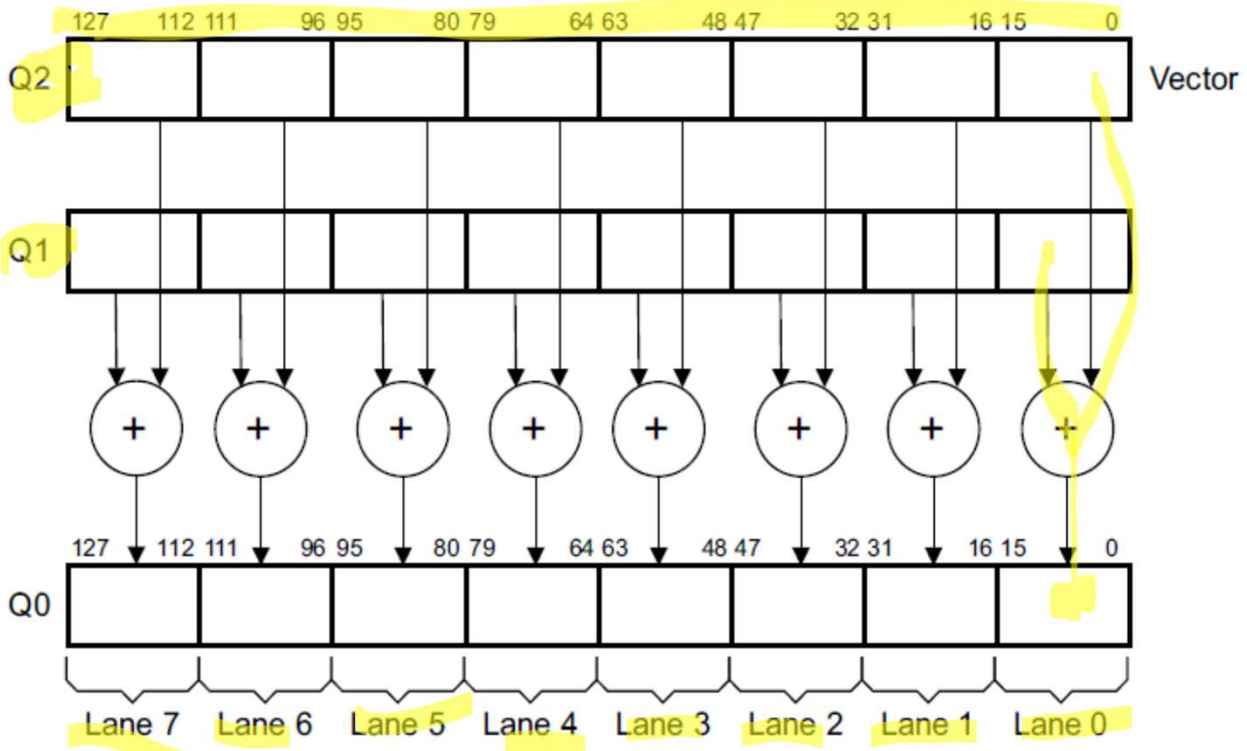
- Array ordering is always from the least significant bit

NEON Register File

- **Extension** register file
- 32 doublewords (64 bits long) D0-D31
- Alternate views possible
  - Word: S0-S31
  - Quadword: Q0-Q15



8-Way 16-bit Integer Add Operation



- `VADD.I16 Q0, Q1, Q2` performs a parallel addition of eight lanes of 16-bit (8 x 16 = 128) integer (I) elements from vectors in Q1 and Q2, storing the result in Q0.

Cortex-A72 ASIMD Instruction Execution

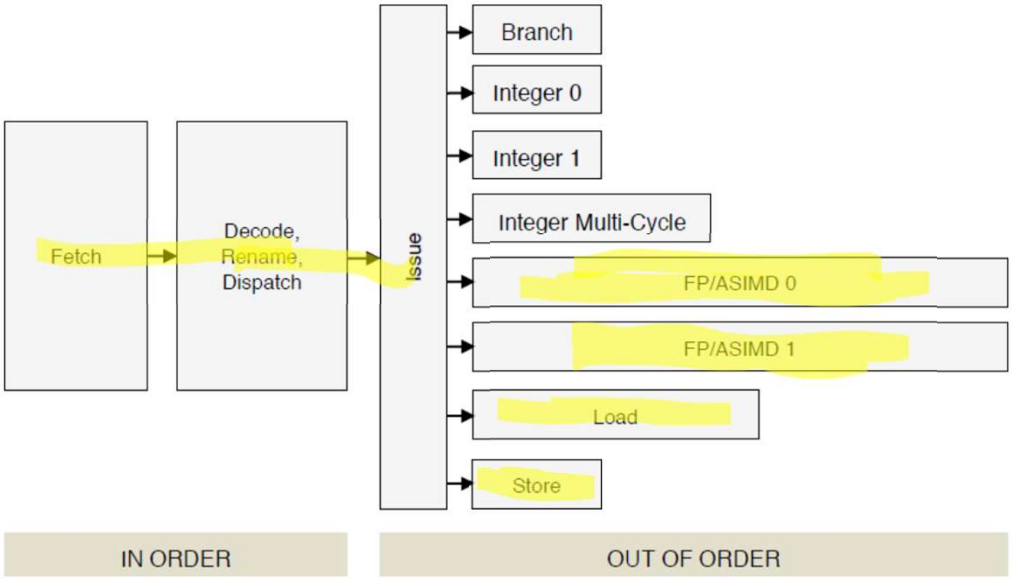
- Up to 4 processing pipelines used for ASIMD instructions
  - FP/ASIMD 0 (F0)
    - ALU, integer multiply, miscellaneous
  - FP/ASIMD 1 (F1)
    - ALU, shift, miscellaneous
  - Load
  - Store
- NEON floating point not fully compliant with IEEE-754

Denormals are flushed to zero

Rounding is fixed to round-to-nearest except for conversion operations

Single precision arithmetic (.F32) only

Separate (scalar) floating-point instructions.



ASIMD Instruction Type	Latency	Thrghpt
ASIMD Integer	3-5	½-2
ASIMD FP	3-7	½-2
ASIMD Load	5-9	½-1
ASIMD Store	1-4	¼-1
ASIMD Misc.	3-8	½-2

ARMV7a SIMD Instruction Syntax

V{<*mod*>}<*op*>{<*shape*>}{<*cond*>}{.<*dt*>} <*dest1*>{, <*dest2*>}, <*src1*>{, <*src2*>}

Shape

	Normal
L	Long
W	Wide
N	Narrow

Condition (with IT instruction)

Modifier

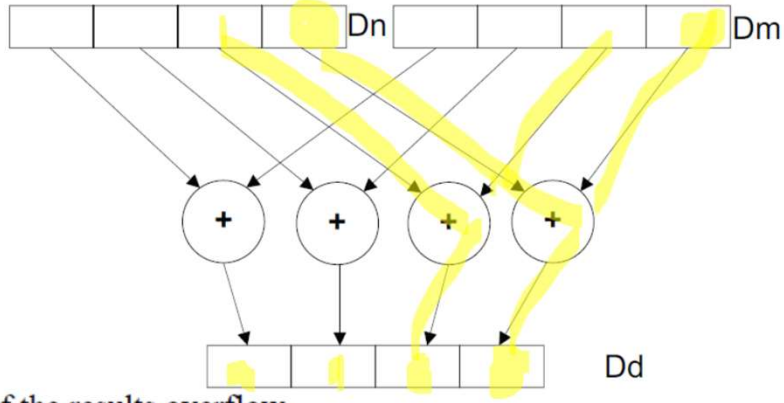
Q	Saturating
H	Halving
D	Doubling before saturation
R	Rounding

Data Type

	8-bit	16-bit	32-bit	64-bit
Unsigned integer	U8	U16	U32	U64
Signed integer	S8	S16	S32	S64
Integer of unspecified type	I8	I16	I32	I64
Floating-point number	not available	F16	F32 or F	not available
Polynomial over {0,1}	P8	P16	not available	not available

Example Instruction: Vector Add

- References:
  - Summary and diagrams from NPG Appendix C
  - Details and intrinsics in NPG Appendix D
- VADD (Vector Add) adds corresponding elements in two vectors, and stores the results in the destination vector.
- *Later:VADD has long, wide and saturating variants.*



Syntax

```
V{Q}ADD{cond}.datatype {Qd,} Qn, Qm
V{Q}ADD{cond}.datatype {Dd,} Dn, Dm
```

where:

cond is an optional conditional code.

Q indicates that saturation is performed if any of the results overflow.

datatype is one of:

- I8, I16, I32, I64, F32 for VADD
- S8, S16, S32 for VQADD, VADDL or VADDW
- U8, U16, U32 for VQADD, VADDL or VADDW
- S64, U64 for VQADD.

Qd, Qn, and Qm specify the destination, first operand and second operand registers for a quadword operation.

Dd, Dn, and Dm specify the destination, first operand and second operand registers for a doubleword operation.

Instruction Types by Operation

- SIMD (lane-by-lane) vector operations
  - vadd, vsub, vmul, vmla, vabs, vabd, vaba
  - vbif, vbit, vbsl,
  - vc*, vac*, vtst
  - vsh*, vneg, vrev
  - vcvt, vcls, vclz, vcnt
  - vand, vbic, veor, vorr, vorn,
  - vrcp, vrce, vrsqrte, vrsqrts,
- Vector reduction
  - vpadd, vpaddl, vpadal, vpmax, vpmin
- Constructing and deconstructing vectors
  - vdup, vmov, vmvn, vext, vmrs, vmsr,
- Load, store vectors
  - vldr, vstr, vldm, vstm, element/structure
- Vector rearrangement
  - vswp, vtrn, vzip, vuzp, generic intrinsics
- Table look-up
  - vtbl, vtbx

Note: Arch. Ref. Manual and ACLE Organize Instructions Differently

- Instruction type (ARM ARM)
  - Memory
  - Move
  - Type Conversion
  - Logical
  - Math
  - Comparison
  - Miscellaneous

- Nature of operation (ACLE)
  - SIMD (lane-by-lane) vector operations
    - Regular
    - Narrowing
    - Widening
  - Constructing and deconstructing vectors
  - Load, store vectors
  - Vector reduction
  - Vector rearrangement
  - Table look-up