SRSRA

Signed Rounding Shift Right and Accumulate (immediate). This instruction reads each vector element in the source SIMD&FP register, right shifts each result by an immediate value, and accumulates the final results with the vector elements of the destination SIMD&FP register. All the values in this instruction are signed integer values. The results are rounded. For truncated results, see *SSRA*.

Depending on the settings in the *CPACR_EL1*, *CPTR_EL2*, and *CPTR_EL3* registers, and the current Security state and Exception level, an attempt to execute the instruction might be trapped.

It has encodings from 2 classes: Scalar and Vector

Scalar

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
0 1	0	1 1 1 1 1 0	!= 0000	immb	0 0	1	1	0	1	Rn	Rd
	U		immh			o1	o0				

SRSRA <V><d>, <V><n>, #<shift>

```
integer d = UInt(Rd);
integer n = UInt(Rn);

if immh<3> != '1' then UNDEFINED;
constant integer esize = 8 << 3;
constant integer datasize = esize;
integer elements = 1;

integer shift = (esize * 2) - UInt(immh:immb);
boolean unsigned = (U == '1');
boolean round = (o1 == '1');
boolean accumulate = (o0 == '1');
```

Vector

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
0	Q	0	0 1 1 1 1 0	!= 0000	immb	0 0	1	1	0	1	Rn	Rd
		U		immh			o1	o0				

SRSRA <Vd>.<T>, <Vn>.<T>, #<shift>

```
integer d = UInt(Rd);
integer n = UInt(Rn);

if immh == '0000' then SEE(asimdimm);
if immh<3>:Q == '10' then UNDEFINED;
constant integer esize = 8 << HighestSetBit(immh);
constant integer datasize = 64 << UInt(Q);
integer elements = datasize DIV esize;
integer shift = (esize * 2) - UInt(immh:immb);
boolean unsigned = (U == '1');
boolean round = (o1 == '1');
boolean accumulate = (o0 == '1');
```

Assembler Symbols

<V>

Is a width specifier, encoded in “immh”:

immh	<V>
0xxx	RESERVED
1xxx	D

<d> Is the number of the SIMD&FP destination register, in the "Rd" field.

<n> Is the number of the first SIMD&FP source register, encoded in the "Rn" field.

<Vd> Is the name of the SIMD&FP destination register, encoded in the "Rd" field.

<T>

Is an arrangement specifier, encoded in “immh:Q”:

immh	Q	<T>
0000	x	SEE Advanced SIMD modified immediate
0001	0	8B
0001	1	16B
001x	0	4H
001x	1	8H
01xx	0	2S
01xx	1	4S
1xxx	0	RESERVED
1xxx	1	2D

<Vn> Is the name of the SIMD&FP source register, encoded in the "Rn" field.

<shift>

For the scalar variant: is the right shift amount, in the range 1 to 64, encoded in “immh:immb”:

immh	<shift>
0xxx	RESERVED
1xxx	(128-UInt(immh:immb))

For the vector variant: is the right shift amount, in the range 1 to the element width in bits, encoded in “immh:immb”:

immh	<shift>
0000	SEE Advanced SIMD modified immediate
0001	(16-UInt(immh:immb))
001x	(32-UInt(immh:immb))
01xx	(64-UInt(immh:immb))
1xxx	(128-UInt(immh:immb))

Operation

```
CheckFPAdvSIMDEnabled64();
bits(datasize) operand = V[n, datasize];
bits(datasize) operand2;
bits(datasize) result;
integer element;

operand2 = if accumulate then V[d, datasize] else Zeros(datasize);
for e = 0 to elements-1
    element = RShr(Int(Elem[operand, e, esize], unsigned), shift, rounc
    Elem[result, e, esize] = Elem[operand2, e, esize] + element<esize-1

V[d, datasize] = result;
```

Base Instructions SIMD&FP Instructions SVE Instructions SME Instructions Index by Encoding Sh Pseu

Internal version only: isa v33.64, AdvSIMD v29.12, pseudocode no_diffs_2023_09_RC2, sve v2023-06_rel ; Build timestamp: 2023-09-18T17:56