SRSHL

Signed Rounding Shift Left (register). This instruction takes each signed integer value in the vector of the first source SIMD&FP register, shifts it by a value from the least significant byte of the corresponding element of the second source SIMD&FP register, places the results in a vector, and writes the vector to the destination SIMD&FP register.

If the shift value is positive, the operation is a left shift. If the shift value is negative, it is a rounding right shift. For a truncating shift, see SSHL.

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 0 size	1	Rm	0 1 0 1 0 1	Rn	Rd
	U				R S		

```
        SRSHL  <V><d>, <V><n>, <V><m>

    integer d = UInt(Rd);
    integer n = UInt(Rn);
    integer m = UInt(Rm);
    constant integer esize = 8 << UInt(size);
    constant integer datasize = esize;
    integer elements = 1;
    boolean unsigned = (U == '1');
    boolean rounding = (R == '1');
    boolean saturating = (S == '1');
    if S == '0' && size != '11' then UNDEFINED;
```

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 0 size	1	Rm	0 1 0 1 0 1	Rn	Rd
	U				R S		

```
        SRSHL  <Vd>.<T>, <Vn>.<T>, <Vm>.<T>

    integer d = UInt(Rd);
    integer n = UInt(Rn);
    integer m = UInt(Rm);
    if size:Q == '110' then UNDEFINED;
    constant integer esize = 8 << UInt(size);
    constant integer datasize = 64 << UInt(Q);
    integer elements = datasize DIV esize;
    boolean unsigned = (U == '1');
    boolean rounding = (R == '1');
    boolean saturating = (S == '1');
```

Assembler Symbols

<V>

Is a width specifier, encoded in "size":

size	<V>
0x	RESERVED
10	RESERVED
11	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.

<m>   Is the number of the second SIMD&FP source register, encoded in the "Rm" field.

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

<T>

Is an arrangement specifier, encoded in "size:Q":

size	Q	<T>
00	0	8B
00	1	16B
01	0	4H
01	1	8H
10	0	2S
10	1	4S
11	0	RESERVED
11	1	2D

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

<Vm>   Is the name of the second SIMD&FP source register, encoded in the "Rm" field.

Operation

```
CheckFPAdvSIMDEnabled64();
bits(datasize) operand1 = V[n, datasize];
bits(datasize) operand2 = V[m, datasize];
bits(datasize) result;

boolean sat;

for e = 0 to elements-1
    integer element = Int(Elem[operand1, e, esize], unsigned);
    integer shift = SInt(Elem[operand2, e, esize]<7:0>);
    if shift >= 0 then // left shift
```

```
            element = element << shift;
        else // right shift
            shift = -shift;
            element = RShr(element, shift, rounding);

        if saturating then
            (Elem[result, e, esize], sat) = SatQ(element, esize, unsigned);
            if sat then FPSR.QC = '1';
        else
            Elem[result, e, esize] = element<esize-1:0>;

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