SQSHLR

Signed saturating shift left reversed vectors (predicated)

Shift active signed elements of the second source vector by corresponding elements of the first source vector and destructively place the results in the corresponding elements of the first source vector. A positive shift amount performs a left shift, otherwise a right shift by the negated shift amount is performed. Each result element is saturated to the N-bit element's signed integer range $-2^{(N-1)}$ to $(2^{(N-1)})-1$. Inactive elements in the destination vector register remain unmodified.

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 0 0 1 0 0	size	0 0 1	1 0	0	1 0 0		Pg	Zm	Zdn
			Q R N	U					

SQSHLR **<Zdn>.<T>**, **<Pg>/M**, **<Zdn>.<T>**, **<Zm>.<T>**

```
if !HaveSVE2() && !HaveSME() then UNDEFINED;
constant integer esize = 8 << UInt(size);
integer g = UInt(Pg);
integer m = UInt(Zm);
integer dn = UInt(Zdn);
```

Assembler Symbols

<Zdn> Is the name of the first source and destination scalable vector register, encoded in the "Zdn" field.

<T> Is the size specifier, encoded in "size":

size	<T>
00	B
01	H
10	S
11	D

<Pg> Is the name of the governing scalable predicate register P0-P7, encoded in the "Pg" field.

<Zm> Is the name of the second source scalable vector register, encoded in the "Zm" field.

Operation

```
CheckSVEEnabled();
constant integer VL = CurrentVL;
constant integer PL = VL DIV 8;
constant integer elements = VL DIV esize;
```

```
    bits(PL) mask = P[g, PL];
    bits(VL) operand1 = if AnyActiveElement(mask, esize) then Z[m, VL] else
    bits(VL) operand2 = Z[dn, VL];
    bits(VL) result;

    for e = 0 to elements-1
        if ActivePredicateElement(mask, e, esize) then
            integer element = SInt(Elem[operand1, e, esize]);
            integer shift = ShiftSat(SInt(Elem[operand2, e, esize]), esize)
            integer res;
            if shift >= 0 then
                res = element << shift;
            else
                shift = -shift;
                res = element >> shift;
            Elem[result, e, esize] = SignedSat(res, esize);
        else
            Elem[result, e, esize] = Elem[operand2, e, esize];

    Z[dn, VL] = result;
```

Operational information

This instruction might be immediately preceded in program order by a
MOVPRFX instruction. The MOVPRFX instruction must conform to all of the
following requirements, otherwise the behavior of the MOVPRFX and this
instruction is unpredictable:

- The MOVPRFX instruction must be unpredicated, or be predicated
  using the same governing predicate register and source element
  size as this instruction.
- The MOVPRFX instruction must specify the same destination register
  as this instruction.
- The destination register must not refer to architectural register
  state referenced by any other source operand register of this
  instruction.

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