UQSHL (immediate)

Unsigned saturating shift left by immediate

Shift left by immediate each active unsigned element of the source vector, and destructively place the results in the corresponding elements of the source vector. Each result element is saturated to the N-bit element's unsigned integer range 0 to (2^N)-1. The immediate shift amount is an unsigned value in the range 0 to number of bits per element minus 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 0 0 0 0 1 0 0	tszh	0 0	0	1	1	1	1	0	0	Pg	tszl	imm3	Zdn
					L	U							

UQSHL **<Zdn>.<T>**, **<Pg>/M**, **<Zdn>.<T>**, **#<const>**

```
if !HaveSVE2() && !HaveSME() then UNDEFINED;
constant bits(4) tsize = tszh:tszl;
if tsize == '0000' then UNDEFINED;
constant integer esize = 8 << HighestSetBit(tsize);
integer g = UInt(Pg);
integer dn = UInt(Zdn);
integer shift = UInt(tsize:imm3) - esize;
```

Assembler Symbols

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

<T>

Is the size specifier, encoded in "tszh:tszl":

tszh	tszl	<T>
00	00	RESERVED
00	01	B
00	1x	H
01	xx	S
1x	xx	D

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

<const>   Is the immediate shift amount, in the range 0 to number of bits per element minus 1, encoded in "tszh:tszl:imm3".

Operation

```
CheckSVEEnabled();
constant integer VL = CurrentVL;
```

```
    constant integer PL = VL DIV 8;
    constant integer elements = VL DIV esize;
    bits(VL) operand1 = Z[dn, VL];
    bits(PL) mask = P[g, PL];
    bits(VL) result;

    for e = 0 to elements-1
        integer element1 = UInt(Elem[operand1, e, esize]);
        if ActivePredicateElement(mask, e, esize) then
            integer res = element1 << shift;
            Elem[result, e, esize] = UnsignedSat(res, esize);
        else
            Elem[result, e, esize] = Elem[operand1, 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.