URSRA

Unsigned rounding shift right and accumulate (immediate)

Shift right by immediate each unsigned element of the source vector, inserting zeroes, and add the rounded intermediate result destructively to the corresponding elements of the addend vector. The immediate shift amount is an unsigned value in the range 1 to number of bits per element. This instruction is unpredicated.

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 1	tszh	0	tszl	imm3	1 1 1 0	1	1	Zn	Zda
						R	U		

```
URSRA <Zda>.<T>, <Zn>.<T>, #<const>
```

```
if !HaveSVE2() && !HaveSME() then UNDEFINED;
bits(4) tsize = tszh:tszl;
if tsize == '0000' then UNDEFINED;
constant integer esize = 8 << HighestSetBit(tsize);
integer n = UInt(Zn);
integer da = UInt(Zda);
integer shift = (2 * esize) – UInt(tsize:imm3);
```

Assembler Symbols

<Zda>   Is the name of the third source and destination scalable vector register, encoded in the "Zda" 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

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

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

Operation

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

```
bits(VL) operand1 = Z[n, VL];
bits(VL) operand2 = Z[da, VL];
bits(VL) result;

for e = 0 to elements-1
    integer element = (UInt(Elem[operand1, e, esize]) + (1 << (shift -
    Elem[result, e, esize] = Elem[operand2, e, esize] + element<esize-1

Z[da, 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.
- 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.