ASR (wide elements, predicated)

Arithmetic shift right by 64-bit wide elements (predicated)

Shift right, preserving the sign bit, active elements of the first source vector by corresponding overlapping 64-bit elements of the second source vector and destructively place the results in the corresponding elements of the first source vector. The shift amount is a vector of unsigned 64-bit doubleword elements in which all bits are significant, and not used modulo the destination element size. 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	size		0	1	1	0	0	0	1	0	0		Pg			Zm				Zdn				

R   L   U

ASR **\<Zdn\>.\<T\>**, **\<Pg\>**/M, **\<Zdn\>.\<T\>**, **\<Zm\>**.D

```
if !HaveSVE() && !HaveSME() then UNDEFINED;
if size == '11' then UNDEFINED;
constant integer esize = 8 << UInt(size);
integer g = UInt(Pg);
integer dn = UInt(Zdn);
integer m = UInt(Zm);
```

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	RESERVED

\<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 = Z[dn, VL];
    bits(VL) operand2 = if AnyActiveElement(mask, esize) then Z[m, VL] else
    bits(VL) result;

    for e = 0 to elements-1
        if ActivePredicateElement(mask, e, esize) then
            bits(esize) element1 = Elem[operand1, e, esize];
            bits(64) element2 = Elem[operand2, (e * esize) DIV 64, 64];
            integer shift = Min(UInt(element2), esize);
            Elem[result, e, esize] = ASR(element1, shift);
        else
            Elem[result, e, esize] = Elem[operand1, e, esize];

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

Operational information

If FEAT_SVE2 is implemented or FEAT_SME is implemented, then if PSTATE.DIT is 1:

- The execution time of this instruction is independent of:
    - The values of the data supplied in any of its operand registers when its governing predicate register contains the same value for each execution.
    - The values of the NZCV flags.
- The response of this instruction to asynchronous exceptions does not vary based on:
    - The values of the data supplied in any of its operand registers when its governing predicate register contains the same value for each execution.
    - The values of the NZCV flags.

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 destination 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.