

## ASRD

Arithmetic shift right for divide by immediate (predicated)

Shift right by immediate, preserving the sign bit, each active element of the source vector, and destructively place the results in the corresponding elements of the source vector. The result rounds toward zero as in a signed division. The immediate shift amount is an unsigned value in the range 1 to number of bits per element. 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	0	0	1	0	0		Pg	tszl	imm3	Zdn									
L																U															

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

```
if !HaveSVE() && !HaveSME() then UNDEFINED;
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 = (2 * esize) - UInt(tsize:imm3);
```

## 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 1 to number of bits per element, encoded in "tszh:tszl:imm3".

## 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) result;

for e = 0 to elements-1
    if ActivePredicateElement(mask, e, esize) then
        integer element1 = SInt(Elem[operand1, e, esize]);
        if element1 < 0 then
            element1 = element1 + ((1 << shift) - 1);
            Elem[result, e, esize] = (element1 >> shift)<esize-1:0>;
        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 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  
Pseud](#)

