MOVPRFX (predicated)

Move prefix (predicated)

The predicated `MOVPRFX` instruction is a hint to hardware that the instruction may be combined with the destructive instruction which follows it in program order to create a single constructive operation. Since it is a hint it is also permitted to be implemented as a discrete vector copy, and the result of executing the pair of instructions with or without combining is identical. The choice of combined versus discrete operation may vary dynamically.

Unless the combination of a constructive operation with merging predication is specifically required, it is strongly recommended that for performance reasons software should prefer to use the zeroing form of predicated `MOVPRFX` or the unpredicated `MOVPRFX` instruction.

Although the operation of the instruction is defined as a simple predicated vector copy, it is required that the prefixed instruction at PC+4 must be an SVE destructive binary or ternary instruction encoding, or a unary operation with merging predication, but excluding other `MOVPRFX` instructions. The prefixed instruction must specify the same predicate register, and have the same maximum element size (ignoring a fixed 64-bit "wide vector" operand), and the same destination vector as the `MOVPRFX` instruction. The prefixed instruction must not use the destination register in any other operand position, even if they have different names but refer to the same architectural register state. Any other use is UNPREDICTABLE.

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	0	0	0	M	0	0	1	Pg			Zn					Zd				

MOVPRFX <Zd>.<T>, <Pg>/<ZM>, <Zn>.<T>

```
if !HaveSVE() && !HaveSME() then UNDEFINED;
constant integer esize = 8 << UInt(size);
integer g = UInt(Pg);
integer n = UInt(Zn);
integer d = UInt(Zd);
boolean merging = (M == '1');
```

Assembler Symbols

<Zd>	Is the name of the destination scalable vector register, encoded in the "Zd" 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 predication qualifier, encoded in “M”:

M	<ZM>
0	Z
1	M

<Zn>

Is the name of the source scalable vector register, encoded in the "Zn" 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[n, VL] else
bits(VL) dest = Z[d, VL];
bits(VL) result;

for e = 0 to elements-1
    if ActivePredicateElement(mask, e, esize) then
        bits(esize) element = Elem[operand1, e, esize];
        Elem[result, e, esize] = element;
    elsif merging then
        Elem[result, e, esize] = Elem[dest, e, esize];
    else
        Elem[result, e, esize] = Zeros(esize);

Z[d, 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.

Internal version only: isa v33.64, AdvSIMD v29.12, pseudocode no_diffs_2023_09_RC2, sve v2023-06_rel ; Build timestamp: 2023-09-18T17:56