SEL

Multi-vector conditionally select elements from two vectors

Read active elements from the two or four first source vectors and inactive elements from the two or four second source vectors and place in the corresponding elements of the two or four destination vectors.
It has encodings from 2 classes: Two registers and Four registers

Two registers
(FEAT_SME2)

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
1	1	0	0	0	0	0	1	size		1	Zm				0	1	0	0	PNg			Zn				0	Zd				0

SEL { <Zd1>.<T>-<Zd2>.<T> }, <PNg>, { <Zn1>.<T>-<Zn2>.<T> }, { <Zm1>

```
if !HaveSME2() then UNDEFINED;
constant integer esize = 8 << UInt(size);
integer n = UInt(Zn:'0');
integer m = UInt(Zm:'0');
integer d = UInt(Zd:'0');
integer g = UInt('1':PNg);
constant integer nreg = 2;
```

Four registers
(FEAT_SME2)

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
1	1	0	0	0	0	0	1	size		1	Zm				0	1	1	0	0	PNg			Zn			0	0	Zd			0	0

SEL { <Zd1>.<T>-<Zd4>.<T> }, <PNg>, { <Zn1>.<T>-<Zn4>.<T> }, { <Zm1>

```
if !HaveSME2() then UNDEFINED;
constant integer esize = 8 << UInt(size);
integer n = UInt(Zn:'00');
integer m = UInt(Zm:'00');
integer d = UInt(Zd:'00');
integer g = UInt('1':PNg);
constant integer nreg = 4;
```

Assembler Symbols

<Zd1> — For the two registers variant: is the name of the first destination scalable vector register of a multi-vector sequence, encoded as "Zd" times 2.

For the four registers variant: is the name of the first destination scalable vector register of a multi-vector sequence, encoded as "Zd" times 4.

<T>

Is the size specifier, encoded in “size”:

size	<T>
00	B
01	H
10	S
11	D

<Zd4> Is the name of the fourth destination scalable vector register of a multi-vector sequence, encoded as "Zd" times 4 plus 3.

<Zd2> Is the name of the second destination scalable vector register of a multi-vector sequence, encoded as "Zd" times 2 plus 1.

<PNg> Is the name of the governing scalable predicate register PN8-PN15, with predicate-as-counter encoding, encoded in the "PNg" field.

<Zn1> For the two registers variant: is the name of the first scalable vector register of a multi-vector sequence, encoded as "Zn" times 2.

For the four registers variant: is the name of the first scalable vector register of a multi-vector sequence, encoded as "Zn" times 4.

<Zn4> Is the name of the fourth scalable vector register of a multi-vector sequence, encoded as "Zn" times 4 plus 3.

<Zn2> Is the name of the second scalable vector register of a multi-vector sequence, encoded as "Zn" times 2 plus 1.

<Zm1> For the two registers variant: is the name of the first scalable vector register of a multi-vector sequence, encoded as "Zm" times 2.

For the four registers variant: is the name of the first scalable vector register of a multi-vector sequence, encoded as "Zm" times 4.

<Zm4> Is the name of the fourth scalable vector register of a multi-vector sequence, encoded as "Zm" times 4 plus 3.

<Zm2> Is the name of the second scalable vector register of a multi-vector sequence, encoded as "Zm" times 2 plus 1.

Operation

```
CheckStreamingSVEEnabled();
constant integer VL = CurrentVL;
constant integer PL = VL DIV 8;
constant integer elements = VL DIV esize;
```

```
array [0..3] of bits(VL) results;
bits(PL) pred = P[g, PL];
bits(PL * nreg) mask = CounterToPredicate(pred<15:0>, PL * nreg);

for r = 0 to nreg-1
    bits(VL) operand1 = Z[n+r, VL];
    bits(VL) operand2 = Z[m+r, VL];
    for e = 0 to elements-1
        if ActivePredicateElement(mask, r * elements + e, esize) then
            Elem[results[r], e, esize] = Elem[operand1, e, esize];
        else
            Elem[results[r], e, esize] = Elem[operand2, e, esize];

for r = 0 to nreg-1
    Z[d+r, VL] = results[r];
```

Operational information

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