PEXT (predicate pair)

Set pair of predicates from predicate-as-counter

Expands the source predicate-as-counter into a four-predicate wide mask and copies two quarters of it into the destination predicate registers.

SVE2
(FEAT_SVE2p1)

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	1	0	0	1	0	1	size		1	0	0	0	0	0	0	1	1	1	0	1	0	i1	PNn			1	Pd			

```
PEXT { <Pd1>.<T>, <Pd2>.<T> }, <PNn>[<imm>]
```

```
if !HaveSME2() && !HaveSVE2p1() then UNDEFINED;
constant integer esize = 8 << UInt(size);
integer n = UInt('1':PNn);
integer d0 = UInt(Pd);
integer d1 = (UInt(Pd) + 1) MOD 16;
integer part = UInt(i1);
```

Assembler Symbols

<Pd1>	Is the name of the first destination scalable predicate register, encoded in the "Pd" field.
<T>	Is the size specifier, encoded in “size”:

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

<Pd2>	Is the name of the second destination scalable predicate register, encoded in the "Pd" field.
<PNn>	Is the name of the first source scalable predicate register PN8-PN15, with predicate-as-counter encoding, encoded in the "PNn" field.
<imm>	Is the element index, in the range 0 to 1, encoded in the "i1" field.

Operation

```
if HaveSVE2p1() then CheckSVEEnabled(); else CheckStreamingSVEEnabled()
constant integer VL = CurrentVL;
constant integer PL = VL DIV 8;
constant integer elements = VL DIV esize;
bits(PL) pred = P[n, PL];
bits(PL*4) mask = CounterToPredicate(pred<15:0>, PL*4);
bits(PL) result0;
bits(PL) result1;
constant integer psize = esize DIV 8;

for e = 0 to elements-1
    bit pbit = PredicateElement(mask, part * 2 * elements + e, esize);
    Elem[result0, e, psize] = ZeroExtend(pbit, psize);

for e = 0 to elements-1
    bit pbit = PredicateElement(mask, part * 2 * elements + elements +
    Elem[result1, e, psize] = ZeroExtend(pbit, psize);

P[d0, PL] = result0;
P[d1, PL] = result1;
```

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 registers.
  - 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 registers.
  - The values of the NZCV flags.
