PMULLB

Polynomial multiply long (bottom)

Polynomial multiply over [0, 1] the corresponding even-numbered elements of the first and second source vectors, and place the results in the overlapping double-width elements of the destination vector. This instruction is unpredicated.

ID_AA64ZFR0_EL1.AES indicates whether the 128-bit element variant is implemented. The 128-bit element variant is illegal when executed in Streaming SVE mode, unless FEAT_SME_FA64 is implemented and enabled.

It has encodings from 2 classes: 16-bit or 64-bit elements and 128-bit element

16-bit or 64-bit elements

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	1	0	0	0	1	0	1	!= 00	0	Zm	0	1	1	0	1	0	Zn	Zd
								size							U	T		

```
PMULLB <Zd>.<T>, <Zn>.<Tb>, <Zm>.<Tb>
```

```
if !HaveSVE2() && !HaveSME() then UNDEFINED;
if size<0> == '0' then UNDEFINED;
constant integer esize = 8 << UInt(size);
integer n = UInt(Zn);
integer m = UInt(Zm);
integer d = UInt(Zd);
```

128-bit element
(FEAT_SVE_PMULL128)

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	1	0	0	0	1	0	1	0	0	0	Zm	0	1	1	0	1	0	Zn	Zd
								size<1>	size<0>							U	T		

```
PMULLB <Zd>.Q, <Zn>.D, <Zm>.D
```

```
if !HaveSVE2PMULL128() then UNDEFINED;
constant integer esize = 128;
integer n = UInt(Zn);
integer m = UInt(Zm);
integer d = UInt(Zd);
```

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<1>”:

size<1>	<T>
0	H
1	D

<Zn>
Is the name of the first source scalable vector register, encoded in the "Zn" field.

<Tb>

Is the size specifier, encoded in “size<1>”:

size<1>	<Tb>
0	B
1	S

<Zm>
Is the name of the second source scalable vector register, encoded in the "Zm" field.

Operation

```
if esize < 128 then CheckSVEEnabled(); else CheckNonStreamingSVEEnablec
constant integer VL = CurrentVL;
constant integer elements = VL DIV esize;
bits(VL) operand1 = Z[n, VL];
bits(VL) operand2 = Z[m, VL];
bits(VL) result;

for e = 0 to elements-1
    bits(esize DIV 2) element1 = Elem[operand1, 2*e + 0, esize DIV 2];
    bits(esize DIV 2) element2 = Elem[operand2, 2*e + 0, esize DIV 2];
    Elem[result, e, esize] = PolynomialMult(element1, element2);

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

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.

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