EXT

Extract vector from pair of vectors

Copy the indexed byte up to the last byte of the first source vector to the bottom of the result vector, then fill the remainder of the result starting from the first byte of the second source vector. The result is placed destructively in the destination and first source vector, or constructively in the destination vector. This instruction is unpredicated.

An index that is greater than or equal to the vector length in bytes is treated as zero, resulting in the first source vector being copied to the result unchanged.

The Destructive encoding of 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. 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.

It has encodings from 2 classes: Constructive and Destructive

Constructive

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	1	0	1	1	imm8h	0 0 0	imm8l	Zn	Zd

EXT <Zd>.B, { <Zn1>.B, <Zn2>.B }, #<imm>

```
if !HaveSVE2() && !HaveSME() then UNDEFINED;
constant integer esize = 8;
integer dst = UInt(Zd);
integer s1 = UInt(Zn);
integer s2 = (s1 + 1) MOD 32;
constant integer position = UInt(imm8h:imm8l) * 8;
```

Destructive

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	1	0	0	1	imm8h	0 0 0	imm8l	Zm	Zdn

EXT <Zdn>.B, <Zdn>.B, <Zm>.B, #<imm>

```
if !HaveSVE() && !HaveSME() then UNDEFINED;
constant integer esize = 8;
integer dst = UInt(Zdn);
integer s1 = dst;
integer s2 = UInt(Zm);
constant integer position = UInt(imm8h:imm8l) * 8;
```

Assembler Symbols

<Zd>	Is the name of the destination scalable vector register, encoded in the "Zd" field.
<Zn1>	Is the name of the first scalable vector register of a multi-vector sequence, encoded in the "Zn" field.
<Zn2>	Is the name of the second scalable vector register of a multi-vector sequence, encoded in the "Zn" field.
<Zdn>	Is the name of the first source and destination scalable vector register, encoded in the "Zdn" field.
<Zm>	Is the name of the second source scalable vector register, encoded in the "Zm" field.
<imm>	Is the unsigned immediate operand, in the range 0 to 255, encoded in the "imm8h:imm8l" fields.

Operation

```
CheckSVEEnabled();
constant integer VL = CurrentVL;
constant integer PL = VL DIV 8;
constant integer elements = VL DIV esize;
bits(VL) operand1 = Z[s1, VL];
bits(VL) operand2 = Z[s2, VL];
bits(VL) result;

bits(VL*2) concat = operand2 : operand1;

if position >= VL then
    result = concat<VL-1:0>;
else
    result = concat<(position+VL)-1:position>;

Z[dst, 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 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