ZIP1, ZIP2 (vectors)

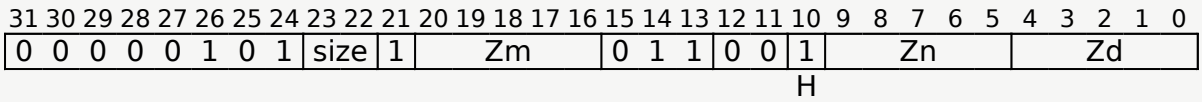
Interleave elements from two half vectors

Interleave alternating elements from the lowest or highest halves of the first and second source vectors and place in elements of the destination vector. This instruction is unpredicated.

The 128-bit element variant requires that the current vector length is at least 256 bits, and if the current vector length is not an integer multiple of 256 bits then the trailing bits are set to zero. ID_AA64ZFR0_EL1.F64MM 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 4 classes: High halves , High halves (quadwords) , Low halves and Low halves (quadwords)

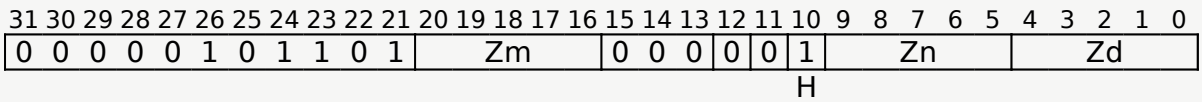
High halves



ZIP2 <Zd>.<T>, <Zn>.<T>, <Zm>.<T>

```
if !HaveSVE() && !HaveSME() then UNDEFINED;
constant integer esize = 8 << UInt(size);
integer n = UInt(Zn);
integer m = UInt(Zm);
integer d = UInt(Zd);
integer part = 1;
```

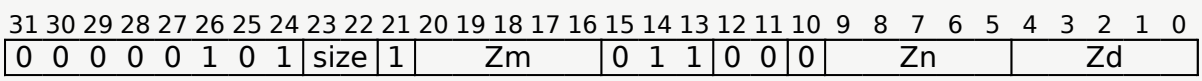
High halves (quadwords)
(FEAT_F64MM)

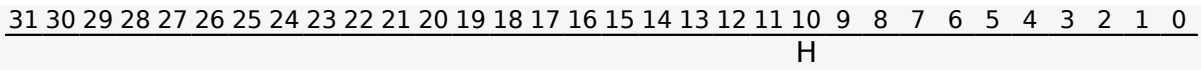


ZIP2 <Zd>.Q, <Zn>.Q, <Zm>.Q

```
if !HaveSVE() || !HaveSVEFP64MatMulExt() then UNDEFINED;
constant integer esize = 128;
integer n = UInt(Zn);
integer m = UInt(Zm);
integer d = UInt(Zd);
integer part = 1;
```

Low halves

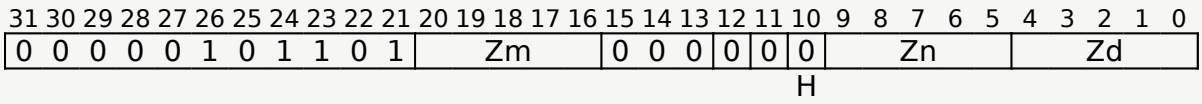




ZIP1 <Zd>.<T>, <Zn>.<T>, <Zm>.<T>

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

Low halves (quadwords)
(FEAT_F64MM)



ZIP1 <Zd>.Q, <Zn>.Q, <Zm>.Q

```
if !HaveSVE() || !HaveSVEFP64MatMulExt() then UNDEFINED;
constant integer esize = 128;
integer n = UInt(Zn);
integer m = UInt(Zm);
integer d = UInt(Zd);
integer part = 0;
```

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

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

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

Operation

```
if esize < 128 then CheckSVEEnabled(); else CheckNonStreamingSVEEnabled
constant integer VL = CurrentVL;
if VL < esize * 2 then UNDEFINED;
constant integer pairs = VL DIV (esize * 2);
```

```
bits(VL) operand1 = Z[n, VL];
bits(VL) operand2 = Z[m, VL];
bits(VL) result = Zeros(VL);

integer base = part * pairs;
for p = 0 to pairs-1
    Elem[result, 2*p+0, esize] = Elem[operand1, base+p, esize];
    Elem[result, 2*p+1, esize] = Elem[operand2, base+p, 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 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