LD1D (scalar plus scalar, strided registers)

Contiguous load of doublewords to multiple strided vectors (scalar index)

Contiguous load of unsigned doublewords to elements of two or four strided vector registers from the memory address generated by a 64-bit scalar base and scalar index which is added to the base address. After each element access the index value is incremented, but the index register is not updated.

Inactive elements will not cause a read from Device memory or signal a fault, and are set to zero in the destination vector.

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 0 1 0 0 0 0 1 0 0 0	Rm	0	1	1	PNg	Rn	T	0	Zt
		msz<1>	msz<0>					N	

        LD1D { **<Zt1>**.D, **<Zt2>**.D }, **<PNg>**/Z, [**<Xn|SP>**, **<Xm>**, LSL #3]

```
if !HaveSME2() then UNDEFINED;
integer n = UInt(Rn);
integer m = UInt(Rm);
integer g = UInt('1':PNg);
constant integer nreg = 2;
integer tstride = 8;
integer t = UInt(T:'0':Zt);
constant integer esize = 64;
```

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 0 1 0 0 0 0 1 0 0 0	Rm	1	1	1	PNg	Rn	T	0	0	Zt
		msz<1>	msz<0>					N		

        LD1D { **<Zt1>**.D, **<Zt2>**.D, **<Zt3>**.D, **<Zt4>**.D }, **<PNg>**/Z, [**<Xn|SP>**, **<Xm**

```
if !HaveSME2() then UNDEFINED;
integer n = UInt(Rn);
integer m = UInt(Rm);
integer g = UInt('1':PNg);
constant integer nreg = 4;
integer tstride = 4;
integer t = UInt(T:'00':Zt);
constant integer esize = 64;
```

Assembler Symbols

<Zt1>

For the two registers variant: is the name of the first scalable vector register Z0-Z7 or Z16-Z23 to be transferred, encoded as "T:'0':Zt".

For the four registers variant: is the name of the first scalable vector register Z0-Z3 or Z16-Z19 to be transferred, encoded as "T:'00':Zt".

<Zt2>

For the two registers variant: is the name of the second scalable vector register Z8-Z15 or Z24-Z31 to be transferred, encoded as "T:'1':Zt".

For the four registers variant: is the name of the second scalable vector register Z4-Z7 or Z20-Z23 to be transferred, encoded as "T:'01':Zt".

<Zt3>

Is the name of the third scalable vector register Z8-Z11 or Z24-Z27 to be transferred, encoded as "T:'10':Zt".

<Zt4>

Is the name of the fourth scalable vector register Z12-Z15 or Z28-Z31 to be transferred, encoded as "T:'11':Zt".

<PNg>

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

<Xn|SP>

Is the 64-bit name of the general-purpose base register or stack pointer, encoded in the "Rn" field.

<Xm>

Is the 64-bit name of the general-purpose offset register, encoded in the "Rm" field.

Operation

```
CheckStreamingSVEEnabled();
constant integer VL = CurrentVL;
constant integer PL = VL DIV 8;
constant integer elements = VL DIV esize;
constant integer mbytes = esize DIV 8;
bits(64) offset;
bits(64) base;
bits(PL) pred = P[g, PL];
bits(PL * nreg) mask = CounterToPredicate(pred<15:0>, PL * nreg);
array [0..3] of bits(VL) values;
boolean contiguous = TRUE;
boolean nontemporal = FALSE;
boolean tagchecked = TRUE;
AccessDescriptor accdesc = CreateAccDescSVE(MemOp_LOAD, nontemporal, co

if !AnyActiveElement(mask, esize) then
    if n == 31 && ConstrainUnpredictableBool(Unpredictable_CHECKSPNONEA
        CheckSPAlignment();
else
    if n == 31 then CheckSPAlignment();
    base = if n == 31 then SP[] else X[n, 64];
```

```
        offset = X[m, 64];

    for r = 0 to nreg-1
        for e = 0 to elements-1
            if ActivePredicateElement(mask, r * elements + e, esize) then
                bits(64) addr = base + (UInt(offset) + r * elements + e) * m
                Elem[values[r], e, esize] = Mem[addr, mbytes, accdesc];
            else
                Elem[values[r], e, esize] = Zeros(esize);

    for r = 0 to nreg-1
        Z[t, VL] = values[r];
        t = t + tstride;
```

Operational information

If PSTATE.DIT is 1, the timing of this instruction is insensitive to the value of the data being loaded or stored when its governing predicate register contains the same value for each execution.