LD1ROD (scalar plus scalar)

Contiguous load and replicate four doublewords (scalar index)

Load four contiguous doublewords to elements of a 256-bit (octaword) vector from the memory address generated by a 64-bit scalar base address and scalar index which is multiplied by 8 and added to the base address.

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

The resulting 256-bit vector is then replicated to fill the destination vector. The instruction 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 in the destination vector are set to zero.

Only the first four predicate elements are used and higher numbered predicate elements are ignored.

ID_AA64ZFR0_EL1.F64MM indicates whether this instruction is implemented.

This instruction is illegal when executed in Streaming SVE mode, unless FEAT_SME_FA64 is implemented and enabled.

SVE
(FEAT_F64MM)

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 1 0	1	1	0 1	Rm	0 0 0	Pg	Rn	Zt
	msz<1>	msz<0>	ssz					

LD1ROD { <Zt>.D }, <Pg>/Z, [<Xn|SP>, <Xm>, LSL #3]

```
if !HaveSVE() || !HaveSVEFP64MatMulExt() then UNDEFINED;
if Rm == '11111' then UNDEFINED;
integer t = UInt(Zt);
integer n = UInt(Rn);
integer m = UInt(Rm);
integer g = UInt(Pg);
constant integer esize = 64;
```

Assembler Symbols

<Zt>	Is the name of the scalable vector register to be transferred, encoded in the "Zt" field.
<Pg>	Is the name of the governing scalable predicate register P0-P7, encoded in the "Pg" 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

```
CheckNonStreamingSVEEnabled();
constant integer VL = CurrentVL;
constant integer PL = VL DIV 8;
if VL < 256 then UNDEFINED;
constant integer elements = 256 DIV esize;
bits(64) base;
bits(PL) mask = P[g, PL]; // low bits only
bits(64) offset;
bits(256) result;
constant integer mbytes = esize DIV 8;
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 e = 0 to elements-1
    if ActivePredicateElement(mask, e, esize) then
        integer eoff = UInt(offset) + e;
        bits(64) addr = base + eoff * mbytes;
        Elem[result, e, esize] = Mem[addr, mbytes, accdesc];
    else
        Elem[result, e, esize] = Zeros(esize);

Z[t, VL] = ZeroExtend(Replicate(result, VL DIV 256), VL);
```

Operational information

If FEAT_SVE2 is implemented or FEAT_SME is implemented, then 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.