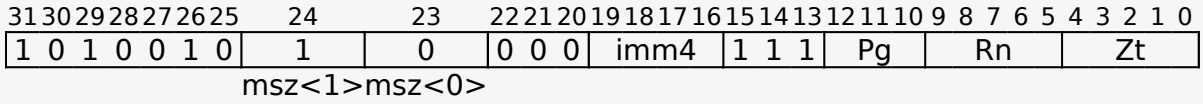
LDNT1W (scalar plus immediate, single register)

Contiguous load non-temporal words to vector (immediate index)

Contiguous load non-temporal of words to elements of a vector register from the memory address generated by a 64-bit scalar base and immediate index in the range -8 to 7 which is multiplied by the vector's in-memory size, irrespective of predication, 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 in the destination vector.

A non-temporal load is a hint to the system that this data is unlikely to be referenced again soon.



LDNT1W { <Zt>.S }, <Pg>/Z, [<Xn|SP>{, #<imm>, MUL VL}]

```
if !HaveSVE() && !HaveSME() then UNDEFINED;
integer t = UInt(Zt);
integer n = UInt(Rn);
integer g = UInt(Pg);
constant integer esize = 32;
integer offset = SInt(imm4);
```

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.
<imm>	Is the optional signed immediate vector offset, in the range -8 to 7, defaulting to 0, encoded in the "imm4" field.

Operation

```
CheckSVEEnabled();
constant integer VL = CurrentVL;
constant integer PL = VL DIV 8;
constant integer elements = VL DIV esize;
bits(64) base;
bits(PL) mask = P[g, PL];
bits(VL) result;
constant integer mbytes = esize DIV 8;
boolean contiguous = TRUE;
```

```
boolean nontemporal = TRUE;
boolean tagchecked = n != 31;
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];

for e = 0 to elements-1
    if ActivePredicateElement(mask, e, esize) then
        integer eoff = (offset * elements) + 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] = result;
```

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.

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