STNT1W (scalar plus scalar, consecutive registers)

Contiguous store non-temporal of words from multiple consecutive vectors (scalar index)

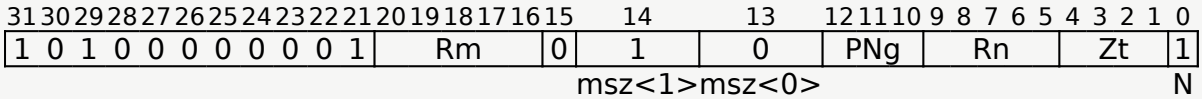
Contiguous store non-temporal of words from elements of two or four consecutive vector registers to the memory address generated by a 64-bit scalar base and scalar index which is added to the base address. After each element access the index value is incremented, but the index register is not updated.

Inactive elements are not written to memory.

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

It has encodings from 2 classes: Two registers and Four registers

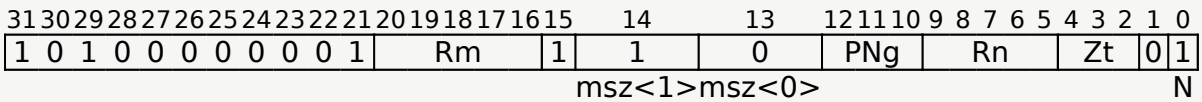
Two registers
(FEAT_SVE2p1)



STNT1W { <Zt1>.S–<Zt2>.S }, <PNg>, [<Xn|SP>, <Xm>, LSL #2]

```
if !HaveSME2() && !HaveSVE2p1() then UNDEFINED;
integer n = UInt(Rn);
integer m = UInt(Rm);
integer g = UInt('1':PNg);
constant integer nreg = 2;
integer t = UInt(Zt:'0');
constant integer esize = 32;
```

Four registers
(FEAT_SVE2p1)



STNT1W { <Zt1>.S–<Zt4>.S }, <PNg>, [<Xn|SP>, <Xm>, LSL #2]

```
if !HaveSME2() && !HaveSVE2p1() then UNDEFINED;
integer n = UInt(Rn);
integer m = UInt(Rm);
integer g = UInt('1':PNg);
constant integer nreg = 4;
integer t = UInt(Zt:'00');
constant integer esize = 32;
```

Assembler Symbols

<Zt1>	For the two registers variant: is the name of the first scalable vector register to be transferred, encoded as "Zt" times 2. For the four registers variant: is the name of the first scalable vector register to be transferred, encoded as "Zt" times 4.
<Zt4>	Is the name of the fourth scalable vector register to be transferred, encoded as "Zt" times 4 plus 3.
<Zt2>	Is the name of the second scalable vector register to be transferred, encoded as "Zt" times 2 plus 1.
<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

```
if HaveSVE2p1() then CheckSVEEnabled(); else 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(VL) src;
bits(PL) pred = P[g, PL];
bits(PL * nreg) mask = CounterToPredicate(pred<15:0>, PL * nreg);
boolean contiguous = TRUE;
boolean nontemporal = TRUE;
boolean tagchecked = TRUE;
AccessDescriptor accdesc = CreateAccDescSVE(MemOp_STORE, nontemporal, c

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
    src = Z[t+r, VL];
    for e = 0 to elements-1
        if ActivePredicateElement(mask, r * elements + e, esize) then
            bits(64) addr = base + (UInt(offset) + r * elements + e) * m
            Mem[addr, mbytes, accdesc] = Elem[src, e, esize];
```

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.

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

Copyright Â© 2010-2023 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.