STNT1H (vector plus scalar)

Scatter store non-temporal halfwords

Scatter store non-temporal of halfwords from the active elements of a vector register to the memory addresses generated by a vector base plus a 64-bit unscaled scalar register offset. 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.

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

It has encodings from 2 classes: [32-bit unscaled offset](#) and [64-bit unscaled offset](#)

32-bit unscaled offset

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 1 1 0 0 1 0	0	1	1 0	Rm	0 0 1	Pg	Zn	Zt
	msz<1>	msz<0>						

STNT1H { **<Zt>**.S }, **<Pg>**, [**<Zn>**.S{, **<Xm>**}]

```
if !HaveSVE2() then UNDEFINED;
integer t = UInt(Zt);
integer n = UInt(Zn);
integer m = UInt(Rm);
integer g = UInt(Pg);
constant integer esize = 32;
constant integer msize = 16;
```

64-bit unscaled offset

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 1 1 0 0 1 0	0	1	0 0	Rm	0 0 1	Pg	Zn	Zt
	msz<1>	msz<0>						

STNT1H { **<Zt>**.D }, **<Pg>**, [**<Zn>**.D{, **<Xm>**}]

```
if !HaveSVE2() then UNDEFINED;
integer t = UInt(Zt);
integer n = UInt(Zn);
integer m = UInt(Rm);
integer g = UInt(Pg);
constant integer esize = 64;
constant integer msize = 16;
```

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.
<Zn>	Is the name of the base scalable vector register, encoded in the "Zn" field.
<Xm>	Is the optional 64-bit name of the general-purpose offset register, defaulting to XZR, encoded in the "Rm" field.

Operation

```
CheckNonStreamingSVEEnabled();
constant integer VL = CurrentVL;
constant integer PL = VL DIV 8;
constant integer elements = VL DIV esize;
bits(PL) mask = P[g, PL];
bits(VL) base;
bits(64) offset;
bits(VL) src;
constant integer mbytes = msize DIV 8;
boolean contiguous = FALSE;
boolean nontemporal = TRUE;
boolean tagchecked = TRUE;
AccessDescriptor accdesc = CreateAccDescSVE(MemOp_STORE, nontemporal, c

if AnyActiveElement(mask, esize) then
    base = Z[n, VL];
    offset = X[m, 64];
    src = Z[t, VL];

for e = 0 to elements-1
    if ActivePredicateElement(mask, e, esize) then
        bits(64) addr = ZeroExtend(Elem[base, e, esize], 64) + offset;
        Mem[addr, mbytes, accdesc] = Elem[src, e, esize]<msize-1:0>;
```

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.