STGM

Store Tag Multiple writes a naturally aligned block of N Allocation Tags, where the size of N is identified in GMID_EL1.BS, and the Allocation Tag written to address A is taken from the source register at 4*A<7:4>+3:4*A<7:4>.

This instruction is undefined at EL0.

This instruction generates an Unchecked access.

Integer
(FEAT_MTE2)

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	0	1	1	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	Xn					Xt				

STGM <Xt>, [<Xn|SP>]

```
if !IsFeatureImplemented(FEAT_MTE2) then UNDEFINED;
integer t = UInt(Xt);
integer n = UInt(Xn);
```

Assembler Symbols

<Xt>	Is the 64-bit name of the general-purpose source register, encoded in the "Xt" field.
<Xn|SP>	Is the 64-bit name of the general-purpose base register or stack pointer, encoded in the "Xn" field.

Operation

```
if PSTATE.EL == EL0 then
    UNDEFINED;

bits(64) data = X[t, 64];
bits(64) address;

if n == 31 then
    CheckSPAlignment();
    address = SP[];
else
    address = X[n, 64];

integer size = 4 * (2 ^ (UInt(GMID_EL1.BS)));
address = Align(address, size);
constant integer count = size >> LOG2_TAG_GRANULE;
integer index = UInt(address<LOG2_TAG_GRANULE+3:LOG2_TAG_GRANULE>);
constant bits(64) curraddress = address;
AccessDescriptor accdesc = CreateAccDescLDGSTG(MemOp_STORE);

for i = 0 to count-1
```

```
    bits(4) tag = Elem[data, index, 4];
    AArch64.MemTag[address, accdesc] = tag;
    address = address + TAG_GRANULE;
    index = index + 1;
```

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