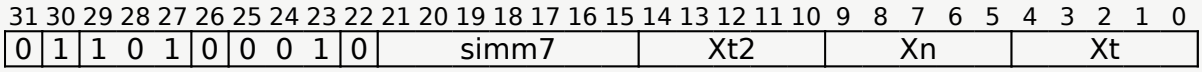
STGP

Store Allocation Tag and Pair of registers stores an Allocation Tag and two 64-bit doublewords to memory, from two registers. The address used for the store is calculated from the base register and an immediate signed offset scaled by the Tag granule. The Allocation Tag is calculated from the Logical Address Tag in the base register.

This instruction generates an Unchecked access.

It has encodings from 3 classes: Post-index , Pre-index and Signed offset

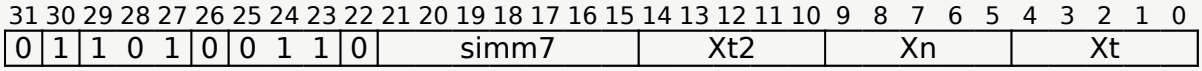
Post-index
(FEAT_MTE)



STGP <Xt1>, <Xt2>, [<Xn|SP>], #<imm>

```
if !IsFeatureImplemented(FEAT_MTE) then UNDEFINED;
integer n = UInt(Xn);
integer t = UInt(Xt);
integer t2 = UInt(Xt2);
bits(64) offset = LSL(SignExtend(simm7, 64), LOG2_TAG_GRANULE);
boolean writeback = TRUE;
boolean postindex = TRUE;
```

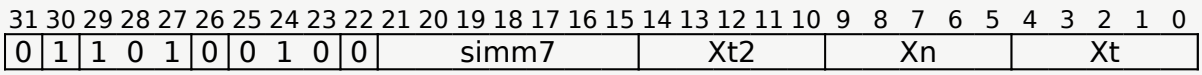
Pre-index
(FEAT_MTE)



STGP <Xt1>, <Xt2>, [<Xn|SP>, #<imm>]!

```
if !IsFeatureImplemented(FEAT_MTE) then UNDEFINED;
integer n = UInt(Xn);
integer t = UInt(Xt);
integer t2 = UInt(Xt2);
bits(64) offset = LSL(SignExtend(simm7, 64), LOG2_TAG_GRANULE);
boolean writeback = TRUE;
boolean postindex = FALSE;
```

Signed offset
(FEAT_MTE)



STGP <Xt1>, <Xt2>, [<Xn|SP>{, #<imm>}]

```
if !IsFeatureImplemented(FEAT_MTE) then UNDEFINED;
integer n = UInt(Xn);
integer t = UInt(Xt);
integer t2 = UInt(Xt2);
bits(64) offset = LSL(SignExtend(simm7, 64), LOG2_TAG_GRANULE);
boolean writeback = FALSE;
boolean postindex = FALSE;
```

Assembler Symbols

<Xt1>	Is the 64-bit name of the first general-purpose register to be transferred, encoded in the "Xt" field.
<Xt2>	Is the 64-bit name of the second general-purpose register to be transferred, encoded in the "Xt2" field.
<Xn|SP>	Is the 64-bit name of the general-purpose base register or stack pointer, encoded in the "Xn" field.
<imm>	For the post-index and pre-index variant: is the signed immediate offset, a multiple of 16 in the range -1024 to 1008, encoded in the "simm7" field. For the signed offset variant: is the optional signed immediate offset, a multiple of 16 in the range -1024 to 1008, defaulting to 0 and encoded in the "simm7" field.

Operation

```
bits(64) address;
bits(64) data1;
bits(64) data2;

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

data1 = X[t, 64];
data2 = X[t2, 64];

if !postindex then
    address = address + offset;

AccessDescriptor accdesc = CreateAccDescLDGSTG(MemOp_STORE);

if !IsAligned(address, TAG_GRANULE) then
    AArch64.Abort(address, AlignmentFault(accdesc));

Mem[address, 8, accdesc] = data1;
Mem[address+8, 8, accdesc] = data2;

AArch64.MemTag[address, accdesc] = AArch64.AllocationTagFromAddress(add
```

```
if writeback then
    if postindex then
        address = address + offset;

    if n == 31 then
        SP[] = address;
    else
        X[n, 64] = address;
```

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