STZ2G

Store Allocation Tags, Zeroing stores an Allocation Tag to two Tag granules of memory, zeroing the associated data locations. 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 source register.

This instruction generates an Unchecked access.

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

Post-index
(FEAT_MTE)

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	1	1	imm9									0	1	Xn					Xt				

STZ2G <Xt|SP>, [<Xn|SP>], #<simm>

```
if !IsFeatureImplemented(FEAT_MTE) then UNDEFINED;
integer n = UInt(Xn);
integer t = UInt(Xt);
bits(64) offset = LSL(SignExtend(imm9, 64), LOG2_TAG_GRANULE);
boolean writeback = TRUE;
boolean postindex = TRUE;
```

Pre-index
(FEAT_MTE)

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	1	1	imm9									1	1	Xn					Xt				

STZ2G <Xt|SP>, [<Xn|SP>, #<simm>]!

```
if !IsFeatureImplemented(FEAT_MTE) then UNDEFINED;
integer n = UInt(Xn);
integer t = UInt(Xt);
bits(64) offset = LSL(SignExtend(imm9, 64), LOG2_TAG_GRANULE);
boolean writeback = TRUE;
boolean postindex = FALSE;
```

Signed offset
(FEAT_MTE)

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	1	1	imm9									1	0	Xn					Xt				

STZ2G <Xt|SP>, [<Xn|SP>{, #<simm>}]

```
if !IsFeatureImplemented(FEAT_MTE) then UNDEFINED;
integer n = UInt(Xn);
```

```
integer t = UInt(Xt);
bits(64) offset = LSL(SignExtend(imm9, 64), LOG2_TAG_GRANULE);
boolean writeback = FALSE;
boolean postindex = FALSE;
```

Assembler Symbols

<Xt|SP>	Is the 64-bit name of the general-purpose source register or stack pointer, 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.
<simm>	Is the optional signed immediate offset, a multiple of 16 in the range -4096 to 4080, defaulting to 0 and encoded in the "imm9" field.

Operation

```
bits(64) address;
bits(64) data = if t == 31 then SP[] else X[t, 64];
bits(4) tag = AArch64.AllocationTagFromAddress(data);

if n == 31 then
    CheckSPAlignment();
    address = SP[];
else
    address = X[n, 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, TAG_GRANULE, accdesc] = Zeros(TAG_GRANULE * 8);
Mem[address+TAG_GRANULE, TAG_GRANULE, accdesc] = Zeros(TAG_GRANULE * 8)

AArch64.MemTag[address, accdesc] = tag;
AArch64.MemTag[address+TAG_GRANULE, accdesc] = tag;

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

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