STR (vector)

Store vector register

Store a vector register to a memory address generated by a 64-bit scalar base, plus an immediate offset in the range -256 to 255 which is multiplied by the current vector register size in bytes. This instruction is unpredicated.

The store is performed as contiguous byte accesses, with no endian conversion and no guarantee of single-copy atomicity larger than a byte. However, if alignment is checked, then the base register must be aligned to 16 bytes.

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	1	1	0	imm9h						0	1	0	imm9l			Rn					Zt				

```
STR <Zt>, [<Xn|SP>{, #<imm>, MUL VL}]
```

```
if !HaveSVE() && !HaveSME() then UNDEFINED;
integer t = UInt(Zt);
integer n = UInt(Rn);
integer imm = SInt(imm9h:imm9l);
```

Assembler Symbols

<Zt>	Is the name of the scalable vector register to be transferred, encoded in the "Zt" field.
<Xn|SP>	Is the 64-bit name of the general-purpose base register or stack pointer, encoded in the "Rn" field.
<imm>	Is the optional signed immediate vector offset, in the range -256 to 255, defaulting to 0, encoded in the "imm9h:imm9l" fields.

Operation

```
CheckSVEEnabled();
constant integer VL = CurrentVL;
constant integer elements = VL DIV 8;
bits(VL) src;
bits(64) base;
integer offset = imm * elements;
boolean contiguous = TRUE;
boolean nontemporal = FALSE;
boolean tagchecked = n != 31;
AccessDescriptor accdesc = CreateAccDescSVE(MemOp_STORE, nontemporal, c

if n == 31 then
    CheckSPAlignment();
    base = SP[];
else
```

```
    base = X[n, 64];

src = Z[t, VL];

boolean aligned = IsAligned(base + offset, 16);

if !aligned && AlignmentEnforced() then
    AArch64.Abort(base + offset, AlignmentFault(accdesc));

for e = 0 to elements-1
    AArch64.MemSingle[base + offset, 1, accdesc, aligned] = Elem[src, e,
    offset = offset + 1;
```

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.

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