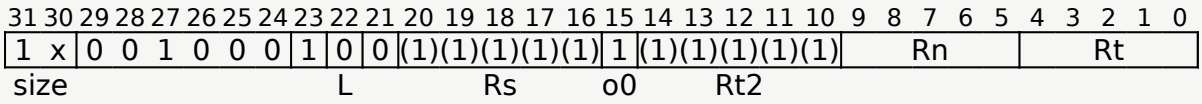
STLR

Store-Release Register stores a 32-bit word or a 64-bit doubleword to a memory location, from a register. The instruction also has memory ordering semantics as described in *Load-Acquire, Store-Release*. For information about memory accesses, see *Load/Store addressing modes*.
It has encodings from 2 classes: No offset and Pre-index

No offset



32-bit (size == 10)

```
STLR <Wt>, [<Xn|SP>{,#0}]
```

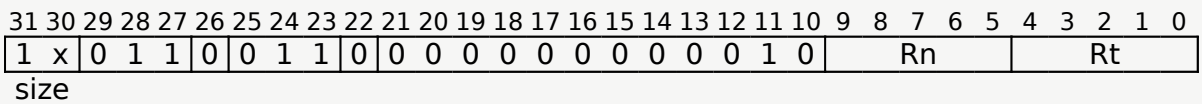
64-bit (size == 11)

```
STLR <Xt>, [<Xn|SP>{,#0}]
```

```
integer n = UInt(Rn);
integer t = UInt(Rt);
boolean wback = FALSE;
integer offset = 0;
boolean rt_unknown = FALSE;

constant integer elsize = 8 << UInt(size);
constant integer datasize = elsize;
boolean tagchecked = n != 31;
```

Pre-index
(FEAT_LRCPC3)



32-bit (size == 10)

```
STLR <Wt>, [<Xn|SP>, #-4]!
```

64-bit (size == 11)

```
STLR <Xt>, [<Xn|SP>, #-8]!
```

```
boolean wback = TRUE;
```

```
integer n = UInt(Rn);
integer t = UInt(Rt);

constant integer datasize = 8 << UInt(size);
integer offset = -1 * (1 << UInt(size));
boolean tagchecked = TRUE;

boolean rt_unknown = FALSE;

if n == t && n != 31 then
    Constraint c = ConstrainUnpredictable(Unpredictable_WBOVERLAPST);
    assert c IN {Constraint_NONE, Constraint_UNKNOWN, Constraint_UNDEF,
    case c of
        when Constraint_NONE    rt_unknown = FALSE;    // value stored
        when Constraint_UNKNOWN rt_unknown = TRUE;    // value stored i
        when Constraint_UNDEF   UNDEFINED;
        when Constraint_NOP     EndOfInstruction();
```

Assembler Symbols

<Wt>	Is the 32-bit name of the general-purpose register to be transferred, encoded in the "Rt" field.
<Xt>	Is the 64-bit name of the general-purpose register to be transferred, encoded in the "Rt" field.
<Xn|SP>	Is the 64-bit name of the general-purpose base register or stack pointer, encoded in the "Rn" field.

Operation

```
bits(64) address;
bits(datasize) data;
constant integer dbytes = datasize DIV 8;

AccessDescriptor accdesc;
accdesc = CreateAccDescAcqRel(MemOp_STORE, tagchecked);

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

address = address + offset;
if rt_unknown then
    data = bits(datasize) UNKNOWN;
else
    data = X[t, datasize];
Mem[address, dbytes, accdesc] = data;

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

Operational information

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