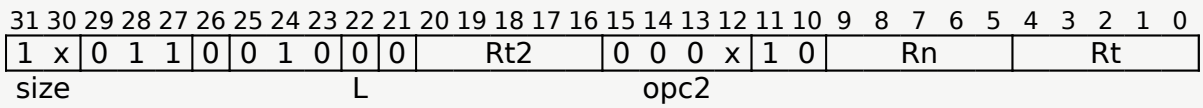
STILP

Store-Release ordered Pair of registers calculates an address from a base register value and an optional offset, and stores two 32-bit words or two 64-bit doublewords to the calculated address, from two registers. For information on single-copy atomicity and alignment requirements, see *Requirements for single-copy atomicity* and *Alignment of data accesses*. The instruction also has memory ordering semantics, as described in *Load-Acquire, Load-AcquirePC, and Store-Release*, with the additional requirement that:

- When using the pre-index addressing mode, the Memory effects associated with Xt2/Wt2 are Ordered-before the Memory effects associated with Xt1/Wt1.
- For all other addressing modes, the Memory effects associated with Xt1/Wt1 are Ordered-before the Memory effects associated with Xt2/Wt2.

For information about memory accesses, see *Load/Store addressing modes*.

Integer
(FEAT_LRCPC3)



32-bit (size == 10 && opc2 == 0001)

```
STILP <Wt1>, <Wt2>, [<Xn|SP>]
```

32-bit pre-index (size == 10 && opc2 == 0000)

```
STILP <Wt1>, <Wt2>, [<Xn|SP>, #-8]!
```

64-bit (size == 11 && opc2 == 0001)

```
STILP <Xt1>, <Xt2>, [<Xn|SP>]
```

64-bit pre-index (size == 11 && opc2 == 0000)

```
STILP <Xt1>, <Xt2>, [<Xn|SP>, #-16]!
```

```
boolean wback;
wback = opc2<0> == '0';
```

STILP has the same constrained unpredictable behavior as STP. For information about this constrained unpredictable behavior, see *Architectural Constraints on UNPREDICTABLE behaviors*, and particularly *STP*.

Assembler Symbols

<Wt1>	Is the 32-bit name of the first general-purpose register to be transferred, encoded in the "Rt" field.
<Wt2>	Is the 32-bit name of the second general-purpose register to be transferred, encoded in the "Rt2" field.
<Xt1>	Is the 64-bit name of the first general-purpose register to be transferred, encoded in the "Rt" field.
<Xt2>	Is the 64-bit name of the second general-purpose register to be transferred, encoded in the "Rt2" field.
<Xn|SP>	Is the 64-bit name of the general-purpose base register or stack pointer, encoded in the "Rn" field.

Shared Decode

```
integer offset;
integer n = UInt(Rn);
integer t = UInt(Rt);
integer t2 = UInt(Rt2);
constant integer scale = 2 + UInt(size<0>);
constant integer datasize = 8 << scale;
offset = if opc2<0> == '0' then -1 * (2 << scale) else 0;

boolean tagchecked = wback || n != 31;

boolean rt_unknown = FALSE;

if wback && (t == n || t2 == n) && 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();
```

Operation

```
bits(64) address;
bits(datasize) data1;
bits(datasize) data2;
constant integer dbytes = datasize DIV 8;

AccessDescriptor accdesc = CreateAccDescAcqRel(MemOp_STORE, tagchecked)

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

```
    else
        address = X[n, 64];

    address = address + offset;

    if rt_unknown && t == n then
        data1 = bits(datasize) UNKNOWN;
    else
        data1 = X[t, datasize];
    if rt_unknown && t2 == n then
        data2 = bits(datasize) UNKNOWN;
    else
        data2 = X[t2, datasize];

    if IsFeatureImplemented(FEAT_LSE2) then
        bits(2*datasize) full_data;
        if BigEndian(accdesc.acctype) then
            full_data = data1:data2;
        else
            full_data = data2:data1;
        accdesc.ispair = TRUE;
        accdesc.highestaddressfirst = offset < 0;
        Mem[address, 2*dbytes, accdesc] = full_data;
    else
        if offset < 0 then
            // Reverse the memory write order for negative pre-index.
            Mem[address+dbytes, dbytes, accdesc] = data2;
            Mem[address, dbytes, accdesc] = data1;
        else
            Mem[address, dbytes, accdesc] = data1;
            Mem[address+dbytes, dbytes, accdesc] = data2;
    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