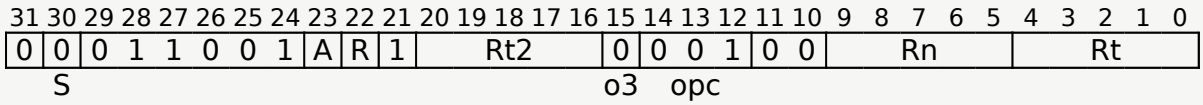
LDCLRP, LDCLRPA, LDCLRPAL, LDCLRPL

Atomic bit clear on quadword in memory atomically loads a 128-bit quadword from memory, performs a bitwise AND with the complement of the value held in a pair of registers on it, and stores the result back to memory. The value initially loaded from memory is returned in the same pair of registers.

- LDCLRPA and LDCLRPAL load from memory with acquire semantics.
- LDCLRPL and LDCLRPAL store to memory with release semantics.
- LDCLRP has neither acquire nor release semantics.

Integer
(FEAT_LSE128)



LDCLRP (A == 0 && R == 0)

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

LDCLRPA (A == 1 && R == 0)

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

LDCLRPAL (A == 1 && R == 1)

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

LDCLRPL (A == 0 && R == 1)

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

```
if !IsFeatureImplemented(FEAT_LSE128) then UNDEFINED;
if Rt == '11111' then UNDEFINED;
if Rt2 == '11111' then UNDEFINED;
integer t = UInt(Rt);
integer t2 = UInt(Rt2);
integer n = UInt(Rn);
boolean acquire = A == '1';
boolean release = R == '1';
boolean tagchecked = n != 31;

boolean rt_unknown = FALSE;

if t == t2 then
```

```
    Constraint c = ConstrainUnpredictable(Unpredictable_LSE128OVERLAP);
    assert c IN {Constraint_UNKNOWN, Constraint_UNDEF, Constraint_NOP};
    case c of
        when Constraint_UNKNOWN rt_unknown = TRUE;    // result is UNKN
        when Constraint_UNDEF   UNDEFINED;
        when Constraint_NOP     EndOfInstruction();
```

Assembler Symbols

<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.

Operation

```
bits(64) address;
bits(64) value1 = X[t, 64];
bits(64) value2 = X[t2, 64];
bits(128) data;
bits(128) store_value;

AccessDescriptor accdesc = CreateAccDescAtomicOp(MemAtomicOp_BIC, acqui

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

store_value = if BigEndian(accdesc.acctype) then value1:value2 else val

bits(128) comparevalue = bits(128) UNKNOWN;    // Irrelevant when not e
data = MemAtomic(address, comparevalue, store_value, accdesc);

if rt_unknown then
    data = bits(128) UNKNOWN;

if BigEndian(accdesc.acctype) then
    X[t, 64] = data<127:64>;
    X[t2, 64] = data<63:0>;
else
    X[t, 64] = data<63:0>;
    X[t2, 64] = data<127:64>;
```

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