LDCLRH, LDCLRAH, LDCLRALH, LDCLRLH

Atomic bit clear on halfword in memory atomically loads a 16-bit halfword from memory, performs a bitwise AND with the complement of the value held in a register on it, and stores the result back to memory. The value initially loaded from memory is returned in the destination register.

- If the destination register is not WZR, `LDCLRAH` and `LDCLRALH` load from memory with acquire semantics.
- `LDCLRLH` and `LDCLRALH` store to memory with release semantics.
- `LDCLRH` has neither acquire nor release semantics.

For more information about memory ordering semantics, see *Load-Acquire, Store-Release*.
For information about memory accesses, see *Load/Store addressing modes*.
This instruction is used by the alias STCLRH, STCLRLH.

Integer
(FEAT_LSE)

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
0	1	1	1	1	0	0	0	A	R	1	Rs					0	0	0	1	0	0	Rn					Rt				
size																	opc														

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

```
LDCLRAH <Ws>, <Wt>, [<Xn|SP>]
```

LDCLRALH (A == 1 && R == 1)

```
LDCLRALH <Ws>, <Wt>, [<Xn|SP>]
```

LDCLRH (A == 0 && R == 0)

```
LDCLRH <Ws>, <Wt>, [<Xn|SP>]
```

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

```
LDCLRLH <Ws>, <Wt>, [<Xn|SP>]
```

```
if !IsFeatureImplemented(FEAT_LSE) then UNDEFINED;

integer t = UInt(Rt);
integer n = UInt(Rn);
integer s = UInt(Rs);

boolean acquire = A == '1' && Rt != '11111';
boolean release = R == '1';
boolean tagchecked = n != 31;
```

Assembler Symbols

<Ws>	Is the 32-bit name of the general-purpose register holding the data value to be operated on with the contents of the memory location, encoded in the "Rs" field.
<Wt>	Is the 32-bit name of the general-purpose register to be loaded, 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.

Alias Conditions

Alias	Is preferred when
STCLRH, STCLRLH	`A == '0' && Rt == '11111'`

Operation

```
bits(64) address;
bits(16) value;
bits(16) data;

AccessDescriptor accdesc = CreateAccDescAtomicOp(MemAtomicOp_BIC, acqui

value = X[s, 16];
if n == 31 then
    CheckSPAlignment();
    address = SP[];
else
    address = X[n, 64];

bits(16) comparevalue = bits(16) UNKNOWN;    // Irrelevant when not exe
data = MemAtomic(address, comparevalue, value, accdesc);

if t != 31 then
    X[t, 32] = ZeroExtend(data, 32);
```

Operational information

If PSTATE.DIT is 1, the timing of this instruction is insensitive to the value of the data being loaded or stored.