CASH, CASAH, CASALH, CASLH

Compare and Swap halfword in memory reads a 16-bit halfword from memory, and compares it against the value held in a first register. If the comparison is equal, the value in a second register is written to memory. If the write is performed, the read and write occur atomically such that no other modification of the memory location can take place between the read and write.

- `CASAH` and `CASALH` load from memory with acquire semantics.
- `CASLH` and `CASALH` store to memory with release semantics.
- `CASH` 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*.

The architecture permits that the data read clears any exclusive monitors associated with that location, even if the compare subsequently fails.

If the instruction generates a synchronous Data Abort, the register which is compared and loaded, that is <Ws>, is restored to the values held in the register before the instruction was executed.

No offset
(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	0	0	1	0	0	0	1	L	1	Rs					o0	1	1	1	1	1	Rn					Rt				
size																															

CASAH (L == 1 && o0 == 0)

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

CASALH (L == 1 && o0 == 1)

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

CASH (L == 0 && o0 == 0)

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

CASLH (L == 0 && o0 == 1)

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

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

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

boolean acquire = L == '1';
boolean release = o0 == '1';
boolean tagchecked = n != 31;
```

Assembler Symbols

<Ws>	Is the 32-bit name of the general-purpose register to be compared and loaded, encoded in the "Rs" field.
<Wt>	Is the 32-bit name of the general-purpose register to be conditionally stored, 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(16) comparevalue;
bits(16) newvalue;
bits(16) data;

AccessDescriptor accdesc = CreateAccDescAtomicOp(MemAtomicOp_CAS, acqui

comparevalue = X[s, 16];
newvalue = X[t, 16];

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

data = MemAtomic(address, comparevalue, newvalue, accdesc);

X[s, 32] = ZeroExtend(data, 32);
```
