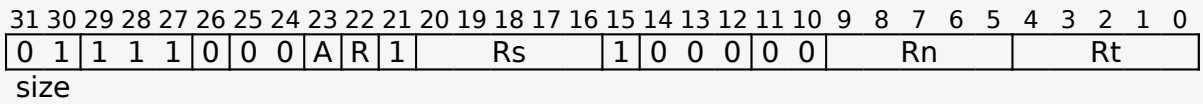
SWPH, SWPAH, SWPALH, SWPLH

Swap halfword in memory atomically loads a 16-bit halfword from a memory location, and stores the value held in a register back to the same memory location. The value initially loaded from memory is returned in the destination register.

- If the destination register is not WZR, `SWPAH` and `SWPALH` load from memory with acquire semantics.
- `SWPLH` and `SWPALH` store to memory with release semantics.
- `SWPH` 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*.

Integer
(FEAT_LSE)



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

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

SWPALH (A == 1 && R == 1)

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

SWPH (A == 0 && R == 0)

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

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

```
SWPLH <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 to be stored, 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.

Operation

```
bits(64) address;
bits(16) data;
bits(16) store_value;
AccessDescriptor accdesc = CreateAccDescAtomicOp(MemAtomicOp_SWP, acqui

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

store_value = X[s, 16];

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

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


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