

SWPP, SWPPA, SWPPAL, SWPPL

Swap quadword in memory atomically loads a 128-bit quadword from a memory location, and stores the value held in a pair of registers back to the same memory location. The value initially loaded from memory is returned in the same pair of registers.

- SWPPA and SWPPAL load from memory with acquire semantics.
- SWPPL and SWPPAL store to memory with release semantics.
- SWPP has neither acquire nor release semantics.

Integer

(FEAT_LSE128)

| 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 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | A | R | 1 | Rt2 | | | | 1 | 0 | 0 | 0 | 0 | 0 | Rn | | | | Rt | | | | | | |
| S | | | | | | | | | | o3 | | | | | | | | | | opc | | | | | | | | | | | |

SWPP (A == 0 && R == 0)

SWPP <Xt1>, <Xt2>, [<Xn|SP>]

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

SWPPA <Xt1>, <Xt2>, [<Xn|SP>]

SWPPAL (A == 1 && R == 1)

SWPPAL <Xt1>, <Xt2>, [<Xn|SP>]

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

SWPPL <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\_SWP, 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.

[Base
Instructions](#)

[SIMD&FP
Instructions](#)

[SVE
Instructions](#)

[SME
Instructions](#)

[Index by
Encoding](#)

[Sh
Pseud](#)

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

Copyright © 2010-2023 Arm Limited or its affiliates. All rights reserved. This
document is Non-Confidential.