

STXR

Store Exclusive Register stores a 32-bit word or a 64-bit doubleword from a register to memory if the PE has exclusive access to the memory address, and returns a status value of 0 if the store was successful, or of 1 if no store was performed. See [Synchronization and semaphores](#). For information about memory accesses, see [Load/Store addressing modes](#).

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
1 x		0 0		1 0		0 0		0 0		0 0		Rs				0		(1)(1)(1)(1)(1)				Rn				Rt					
size		L										o0				Rt2															

32-bit (size == 10)

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

64-bit (size == 11)

```
STXR <Ws>, <Xt>, [<Xn|SP>{, #0}]
```

```
integer n = UInt(Rn);
integer t = UInt(Rt);
integer s = UInt(Rs);    // ignored by all loads and store-release

constant integer elsize = 8 << UInt(size);
boolean tagchecked = n != 31;

boolean rt_unknown = FALSE;
boolean rn_unknown = FALSE;
if s == t then
    Constraint c = ConstrainUnpredictable(Unpredictable_DATAOVERLAP);
    assert c IN {Constraint_UNKNOWN, Constraint_UNDEF, Constraint_NOP};
    case c of
        when Constraint_UNKNOWN rt_unknown = TRUE;    // store UNKNOWN
        when Constraint_UNDEF    UNDEFINED;
        when Constraint_NOP      EndOfInstruction();
if s == n && n != 31 then
    Constraint c = ConstrainUnpredictable(Unpredictable_BASEOVERLAP);
    assert c IN {Constraint_UNKNOWN, Constraint_UNDEF, Constraint_NOP};
    case c of
        when Constraint_UNKNOWN rn_unknown = TRUE;    // address is UNK
        when Constraint_UNDEF    UNDEFINED;
        when Constraint_NOP      EndOfInstruction();
```

For information about the constrained unpredictable behavior of this instruction, see [Architectural Constraints on UNPREDICTABLE behaviors](#), and particularly [STXR](#).

Assembler Symbols

<Ws>	Is the 32-bit name of the general-purpose register into which the status result of the store exclusive is written, encoded in the "Rs" field. The value returned is: 0 If the operation updates memory. 1 If the operation fails to update memory.
<Xt>	Is the 64-bit name of the general-purpose register to be transferred, encoded in the "Rt" field.
<Wt>	Is the 32-bit name of the general-purpose register to be transferred, 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.

Aborts and alignment

If a synchronous Data Abort exception is generated by the execution of this instruction:

- Memory is not updated.
- <Ws> is not updated.

Accessing an address that is not aligned to the size of the data being accessed causes an Alignment fault Data Abort exception to be generated, subject to the following rules:

- If AArch64.ExclusiveMonitorsPass() returns TRUE, the exception is generated.
- Otherwise, it is implementation defined whether the exception is generated.

If AArch64.ExclusiveMonitorsPass() returns FALSE and the memory address, if accessed, would generate a synchronous Data Abort exception, it is implementation defined whether the exception is generated.

Operation

```
bits(64) address;  
bits(elsize) data;  
constant integer dbytes = elsize DIV 8;  
  
AccessDescriptor accdesc = CreateAccDescExLDST(MemOp_STORE, FALSE, tagc  
  
if n == 31 then  
    CheckSPAlignment();  
    address = SP[];  
elseif rn_unknown then  
    address = bits(64) UNKNOWN;  
else
```

```

        address = X[n, 64];

    if rt_unknown then
        data = bits(elsize) UNKNOWN;
    else
        data = X[t, elsize];

    bit status = '1';
    // Check whether the Exclusives monitors are set to include the
    // physical memory locations corresponding to virtual address
    // range [address, address+dbytes-1].

    // If AArch64.ExclusiveMonitorsPass() returns FALSE and the memory address
    // if accessed, would generate a synchronous Data Abort exception, it is
    // IMPLEMENTATION DEFINED whether the exception is generated.
    // It is a limitation of this model that synchronous Data Aborts are not
    // generated in this case, as Mem[] is not called.
    // If FEAT_SPE is implemented, it is also IMPLEMENTATION DEFINED whether
    // physical address packet is output when permitted and when
    // AArch64.ExclusiveMonitorPass() returns FALSE for a Store Exclusive instruction.
    // This behavior is not reflected here due to the previously stated limitation.
    if AArch64.ExclusiveMonitorsPass(address, dbytes, accdesc) then
        // This atomic write will be rejected if it does not refer
        // to the same physical locations after address translation.
        Mem[address, dbytes, accdesc] = data;
        status = ExclusiveMonitorsStatus();
    X[s, 32] = ZeroExtend(status, 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.

[Base
Instructions](#)

[SIMD&FP
Instructions](#)

[SVE
Instructions](#)

[SME
Instructions](#)

[Index by
Encoding](#)

[Sh
Pseudocode](#)

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