## LDSMAX, LDSMAXA, LDSMAXAL, LDSMAXL

Atomic signed maximum on word or doubleword in memory atomically loads a 32-bit word or 64-bit doubleword from memory, compares it against the value held in a register, and stores the larger value back to memory, treating the values as signed numbers. The value initially loaded from memory is returned in the destination register.

- If the destination register is not one of WZR or XZR, `LDSMAXA` and `LDSMAXAL` load from memory with acquire semantics.
- `LDSMAXL` and `LDSMAXAL` store to memory with release semantics.
- `LDSMAX` 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 STSMAX, STSMAXL.

**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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | x | 1 | 1 | 1 | 0 | 0 | 0 | A | R | 1 | Rs | | | | | 0 | 1 | 0 | 0 | 0 | 0 | Rn | | | | | Rt | | | | |
| size | | | | | | | | | | | | | | | | opc | | | | | | | | | | | | | | | |

**32-bit LDSMAX (size == 10 && A == 0 && R == 0)**

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

**32-bit LDSMAXA (size == 10 && A == 1 && R == 0)**

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

**32-bit LDSMAXAL (size == 10 && A == 1 && R == 1)**

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

**32-bit LDSMAXL (size == 10 && A == 0 && R == 1)**

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

**64-bit LDSMAX (size == 11 && A == 0 && R == 0)**

```
LDSMAX <Xs>, <Xt>, [<Xn|SP>]
```

**64-bit LDSMAXA (size == 11 && A == 1 && R == 0)**

```
LDSMAXA <Xs>, <Xt>, [<Xn|SP>]
```

**64-bit LDSMAXAL (size == 11 && A == 1 && R == 1)**

```
LDSMAXAL <Xs>, <Xt>, [<Xn|SP>]
```

**64-bit LDSMAXL (size == 11 && A == 0 && R == 1)**

```
LDSMAXL <Xs>, <Xt>, [<Xn|SP>]
```

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

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

constant integer datasize = 8 << UInt(size);
integer regsize = if datasize == 64 then 64 else 32;
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. |
| <Xs> | Is the 64-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. |
| <Xt> | Is the 64-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 |
|---|---|
| STSMAX, STSMAXL | `A == '0' && Rt == '11111'` |

**Operation**

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

```
AccessDescriptor accdesc = CreateAccDescAtomicOp(MemAtomicOp_SMAX, acqu

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

bits(datasize) comparevalue = bits(datasize) UNKNOWN;    // Irrelevant
data = MemAtomic(address, comparevalue, value, accdesc);

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

**Operational information**

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

---



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