LDTRSB

Load Register Signed Byte (unprivileged) loads a byte from memory, sign-extends it to 32 bits or 64 bits, and writes the result to a register. The address that is used for the load is calculated from a base register and an immediate offset.

Memory accesses made by the instruction behave as if the instruction was executed at EL0 if the *Effective value* of PSTATE.UAO is 0 and either:

- The instruction is executed at EL1.
- The instruction is executed at EL2 when the *Effective value* of HCR_EL2.{E2H, TGE} is {1, 1}.

Otherwise, the memory access operates with the restrictions determined by the Exception level at which the instruction is executed. 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
0 0	1 1 1	0	0 0	1 x	0	imm9	1 0	Rn	Rt
size			opc						

32-bit (opc == 11)

        LDTRSB <Wt>, [<Xn|SP>{, #<simm>}]

64-bit (opc == 10)

        LDTRSB <Xt>, [<Xn|SP>{, #<simm>}]

    bits(64) offset = SignExtend(imm9, 64);

Assembler Symbols

<Wt>	Is the 32-bit name of the general-purpose register to be transferred, encoded in the "Rt" field.
<Xt>	Is the 64-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.
<simm>	Is the optional signed immediate byte offset, in the range -256 to 255, defaulting to 0 and encoded in the "imm9" field.

Shared Decode

```
integer n = UInt(Rn);
integer t = UInt(Rt);
MemOp memop;
boolean signed;
integer regsize;

if opc<1> == '0' then
    // store or zero-extending load
    memop = if opc<0> == '1' then MemOp_LOAD else MemOp_STORE;
    regsize = 32;
    signed = FALSE;
else
    // sign-extending load
    memop = MemOp_LOAD;
    regsize = if opc<0> == '1' then 32 else 64;
    signed = TRUE;

boolean tagchecked = memop != MemOp_PREFETCH && (n != 31);
```

Operation

```
bits(64) address;
bits(8) data;

boolean privileged = AArch64.IsUnprivAccessPriv();
AccessDescriptor accdesc = CreateAccDescGPR(memop, FALSE, privileged, t

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

address = address + offset;

case memop of
    when MemOp_STORE
        data = X[t, 8];
        Mem[address, 1, accdesc] = data;

    when MemOp_LOAD
        data = Mem[address, 1, accdesc];
        if signed then
            X[t, regsize] = SignExtend(data, regsize);
        else
            X[t, regsize] = ZeroExtend(data, regsize);

    when MemOp_PREFETCH
        Prefetch(address, t<4:0>);
```

Operational information

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