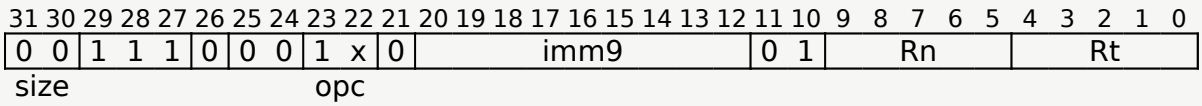
LDRSB (immediate)

Load Register Signed Byte (immediate) loads a byte from memory, sign-extends it to either 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. For information about memory accesses, see *Load/Store addressing modes*.

It has encodings from 3 classes: Post-index , Pre-index and Unsigned offset

Post-index



32-bit (opc == 11)

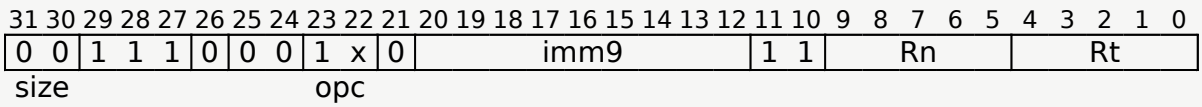
```
LDRSB <Wt>, [<Xn|SP>], #<simm>
```

64-bit (opc == 10)

```
LDRSB <Xt>, [<Xn|SP>], #<simm>
```

```
boolean wback = TRUE;
boolean postindex = TRUE;
bits(64) offset = SignExtend(imm9, 64);
```

Pre-index



32-bit (opc == 11)

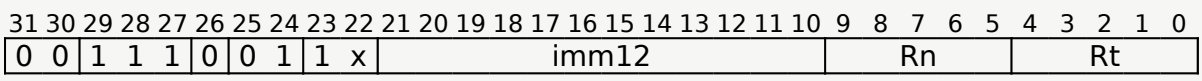
```
LDRSB <Wt>, [<Xn|SP>, #<simm>]!
```

64-bit (opc == 10)

```
LDRSB <Xt>, [<Xn|SP>, #<simm>]!
```

```
boolean wback = TRUE;
boolean postindex = FALSE;
bits(64) offset = SignExtend(imm9, 64);
```

Unsigned offset



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
size								opc																							

32-bit (opc == 11)

```
LDRSB <Wt>, [<Xn|SP>{, #<pimm>}]
```

64-bit (opc == 10)

```
LDRSB <Xt>, [<Xn|SP>{, #<pimm>}]
```

```
boolean wback = FALSE;
boolean postindex = FALSE;
bits(64) offset = LSL(ZeroExtend(imm12, 64), 0);
```

For information about the constrained unpredictable behavior of this instruction, see *Architectural Constraints on UNPREDICTABLE behaviors*, and particularly *LDRSB (immediate)*.

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 signed immediate byte offset, in the range -256 to 255, encoded in the "imm9" field.
<pimm>	Is the optional positive immediate byte offset, in the range 0 to 4095, defaulting to 0 and encoded in the "imm12" 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 && (wback || n != 31);

boolean wb_unknown = FALSE;
boolean rt_unknown = FALSE;
Constraint c;

if memop == MemOp_LOAD && wback && n == t && n != 31 then
    c = ConstrainUnpredictable(Unpredictable_WBOVERLAPLD);
    assert c IN {Constraint_WBSUPPRESS, Constraint_UNKNOWN, Constraint_
    case c of
        when Constraint_WBSUPPRESS wback = FALSE;    // writeback is su
        when Constraint_UNKNOWN    wb_unknown = TRUE;    // writeback i
        when Constraint_UNDEF      UNDEFINED;
        when Constraint_NOP        EndOfInstruction();

if memop == MemOp_STORE && wback && n == t && n != 31 then
    c = ConstrainUnpredictable(Unpredictable_WBOVERLAPST);
    assert c IN {Constraint_NONE, Constraint_UNKNOWN, Constraint_UNDEF,
    case c of
        when Constraint_NONE    rt_unknown = FALSE;    // value stored
        when Constraint_UNKNOWN rt_unknown = TRUE;    // value stored i
        when Constraint_UNDEF   UNDEFINED;
        when Constraint_NOP     EndOfInstruction();
```

Operation

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

boolean privileged = PSTATE.EL != EL0;
AccessDescriptor accdesc = CreateAccDescGPR(memop, FALSE, privileged, t

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

if !postindex then
    address = address + offset;

case memop of
    when MemOp_STORE
        if rt_unknown then
            data = bits(8) UNKNOWN;
        else
            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>);
```

```
if wback then
    if wb_unknown then
        address = bits(64) UNKNOWN;
    elsif postindex then
        address = address + offset;
    if n == 31 then
        SP[] = address;
    else
        X[n, 64] = address;
```

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