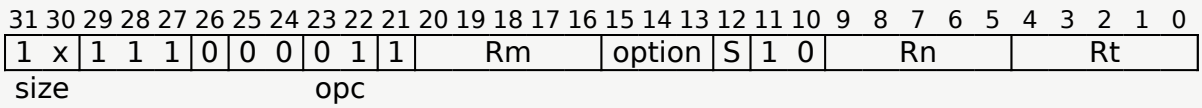
LDR (register)

Load Register (register) calculates an address from a base register value and an offset register value, loads a word from memory, and writes it to a register. The offset register value can optionally be shifted and extended. For information about memory accesses, see *Load/Store addressing modes*.



32-bit (size == 10)

```
LDR <Wt>, [<Xn|SP>, (<Wm>|<Xm>){, <extend> {<amount>}}]
```

64-bit (size == 11)

```
LDR <Xt>, [<Xn|SP>, (<Wm>|<Xm>){, <extend> {<amount>}}]
```

```
integer scale = UInt(size);
if option<1> == '0' then UNDEFINED;    // sub-word index
ExtendType extend_type = DecodeRegExtend(option);
integer shift = if S == '1' then scale else 0;
```

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.
<Wm>	When option<0> is set to 0, is the 32-bit name of the general-purpose index register, encoded in the "Rm" field.
<Xm>	When option<0> is set to 1, is the 64-bit name of the general-purpose index register, encoded in the "Rm" field.

<extend>

Is the index extend/shift specifier, defaulting to LSL, and which must be omitted for the LSL option when <amount> is omitted. encoded in “option”:

option	<extend>
010	UXTW
011	LSL
110	SXTW
111	SXTX

<amount>

For the 32-bit variant: is the index shift amount, optional only when <extend> is not LSL. Where it is permitted to be optional, it defaults to #0. It is encoded in “S”:

S	<amount>
0	#0
1	#2

For the 64-bit variant: is the index shift amount, optional only when <extend> is not LSL. Where it is permitted to be optional, it defaults to #0. It is encoded in “S”:

S	<amount>
0	#0
1	#3

Shared Decode

```
integer n = UInt(Rn);
integer t = UInt(Rt);
integer m = UInt(Rm);
integer regsize;

regsize = if size == '11' then 64 else 32;
constant integer datasize = 8 << scale;
```

Operation

```
bits(64) offset = ExtendReg(m, extend_type, shift, 64);
bits(64) address;
bits(datasize) data;

boolean privileged = PSTATE.EL != EL0;
AccessDescriptor accdesc = CreateAccDescGPR(MemOp_LOAD, FALSE, privileg

if n == 31 then
```

```
    CheckSPAlignment();
    address = SP[];
else
    address = X[n, 64];

address = address + offset;

data = Mem[address, datasize DIV 8, accdesc];
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