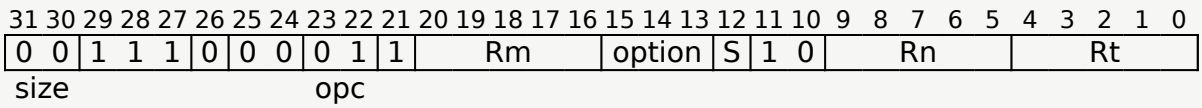
LDRB (register)

Load Register Byte (register) calculates an address from a base register value and an offset register value, loads a byte from memory, zero-extends it, and writes it to a register. For information about memory accesses, see *Load/Store addressing modes*.



Extended register (option != 011)

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

Shifted register (option == 011)

```
LDRB <Wt>, [<Xn|SP>, <Xm>{, LSL <amount>}]
```

```
if option<1> == '0' then UNDEFINED;    // sub-word index
ExtendType extend_type = DecodeRegExtend(option);
```

Assembler Symbols

<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.
<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 specifier, encoded in “option”:

option	<extend>
010	UXTW
110	SXTW
111	SXTX

<amount>	Is the index shift amount, it must be #0, encoded in "S" as 0 if omitted, or as 1 if present.

Shared Decode

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

Operation

```
bits(64) offset = ExtendReg(m, extend_type, 0, 64);
bits(64) address;
bits(8) 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, 1, accdesc];
X[t, 32] = ZeroExtend(data, 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.

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