

## LDR (immediate)

Load Register (immediate) loads a word or doubleword from memory and writes it 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](#). The Unsigned offset variant scales the immediate offset value by the size of the value accessed before adding it to the base register value.

It has encodings from 3 classes: [Post-index](#) , [Pre-index](#) and [Unsigned offset](#)

### Post-index

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		0 1 0		imm9										0 1		Rn				Rt											
size										opc																									

### 32-bit (size == 10)

```
LDR <Wt>, [<Xn|SP>], #<sim>
```

### 64-bit (size == 11)

```
LDR <Xt>, [<Xn|SP>], #<sim>
```

```
boolean wback = TRUE;  
boolean postindex = TRUE;  
integer scale = UInt(size);  
bits(64) offset = SignExtend(imm9, 64);
```

### Pre-index

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	0	1	0	imm9									1	1	Rn				Rt									
size										opc																									

### 32-bit (size == 10)

```
LDR <Wt>, [<Xn|SP>, #<sim>]!
```

### 64-bit (size == 11)

```
LDR <Xt>, [<Xn|SP>, #<sim>]!
```

```
boolean wback = TRUE;  
boolean postindex = FALSE;  
integer scale = UInt(size);  
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								
1		x		1		1		1		0		0		1		0		1		imm12												Rn				Rt			
size										opc																													

### 32-bit (size == 10)

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

### 64-bit (size == 11)

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

```
boolean wback = FALSE;
boolean postindex = FALSE;
integer scale = UInt(size);
bits(64) offset = LSL(ZeroExtend(imm12, 64), scale);
```

For information about the constrained unpredictable behavior of this instruction, see [Architectural Constraints on UNPREDICTABLE behaviors](#), and particularly *LDR (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.
<sim>	Is the signed immediate byte offset, in the range -256 to 255, encoded in the "imm9" field.
<pimm>	For the 32-bit variant: is the optional positive immediate byte offset, a multiple of 4 in the range 0 to 16380, defaulting to 0 and encoded in the "imm12" field as <pimm>/4.  For the 64-bit variant: is the optional positive immediate byte offset, a multiple of 8 in the range 0 to 32760, defaulting to 0 and encoded in the "imm12" field as <pimm>/8.

### Shared Decode

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

regsize = if size == '11' then 64 else 32;
constant integer datasize = 8 << scale;
boolean tagchecked = wback || n != 31;
```

```

boolean wb_unknown = FALSE;
Constraint c;

if wback && n == t && n != 31 then
    c = ConstraintUnpredictable(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();

```

## Operation

```

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];

if !postindex then
    address = address + offset;

data = Mem[address, datasize DIV 8, accdesc];
X[t, regsize] = ZeroExtend(data, regsize);

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.

[Base  
Instructions](#)

[SIMD&FP  
Instructions](#)

[SVE  
Instructions](#)

[SME  
Instructions](#)

[Index by  
Encoding](#)

[Sh  
Pseud](#)

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

Copyright Â© 2010-2023 Arm Limited or its affiliates. All rights reserved. This  
document is Non-Confidential.