

LD4R

Load single 4-element structure and Replicate to all lanes of four registers. This instruction loads a 4-element structure from memory and replicates the structure to all the lanes of the four SIMD&FP registers.

Depending on the settings in the [CPACR_EL1](#), [CPTR_EL2](#), and [CPTR_EL3](#) registers, and the current Security state and Exception level, an attempt to execute the instruction might be trapped.

It has encodings from 2 classes: [No offset](#) and [Post-index](#)

No 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
0	Q	0	0	1	1	0	1	0	1	1	0	0	0	0	0	1	1	1	0	size	Rn			Rt							
L R										o2 opcode S																					

LD4R { [<Vt>.<T>](#), [<Vt2>.<T>](#), [<Vt3>.<T>](#), [<Vt4>.<T>](#) }, [[<Xn|SP>](#)]

```
integer t = UInt(Rt);
integer n = UInt(Rn);
integer m = integer UNKNOWN;
boolean wback = FALSE;
boolean nontemporal = FALSE;
boolean tagchecked = wback || n != 31;
```

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
0	Q	0	0	1	1	0	1	1	1	1	Rm				1	1	1	0	size	Rn				Rt							
L R										opcode S																					

Immediate offset (Rm == 11111)

LD4R { [<Vt>.<T>](#), [<Vt2>.<T>](#), [<Vt3>.<T>](#), [<Vt4>.<T>](#) }, [[<Xn|SP>](#)], [<imm>](#)

Register offset (Rm != 11111)

LD4R { [<Vt>.<T>](#), [<Vt2>.<T>](#), [<Vt3>.<T>](#), [<Vt4>.<T>](#) }, [[<Xn|SP>](#)], [<Xm>](#)

```
integer t = UInt(Rt);
integer n = UInt(Rn);
integer m = UInt(Rm);
boolean wback = TRUE;
boolean nontemporal = FALSE;
boolean tagchecked = wback || n != 31;
```

Assembler Symbols

<Vt> Is the name of the first or only SIMD&FP register to be transferred, encoded in the "Rt" field.

<T> Is an arrangement specifier, encoded in "size:Q":

size	Q	<T>
00	0	8B
00	1	16B
01	0	4H
01	1	8H
10	0	2S
10	1	4S
11	0	1D
11	1	2D

<Vt2> Is the name of the second SIMD&FP register to be transferred, encoded as "Rt" plus 1 modulo 32.

<Vt3> Is the name of the third SIMD&FP register to be transferred, encoded as "Rt" plus 2 modulo 32.

<Vt4> Is the name of the fourth SIMD&FP register to be transferred, encoded as "Rt" plus 3 modulo 32.

<Xn|SP> Is the 64-bit name of the general-purpose base register or stack pointer, encoded in the "Rn" field.

<imm> Is the post-index immediate offset, encoded in "size":

size	<imm>
00	#4
01	#8
10	#16
11	#32

<Xm> Is the 64-bit name of the general-purpose post-index register, excluding XZR, encoded in the "Rm" field.

Shared Decode

```
bits(2) scale = opcode<2:1>;
integer selem = UInt(opcode<0>:R) + 1;
boolean replicate = FALSE;
integer index;

case scale of
  when '11'
    // load and replicate
    if L == '0' || S == '1' then UNDEFINED;
    scale = size;
```

```

        replicate = TRUE;
when '00'
    index = UInt(Q:S:size);    // B[0-15]
when '01'
    if size<0> == '1' then UNDEFINED;
    index = UInt(Q:S:size<1>);    // H[0-7]
when '10'
    if size<1> == '1' then UNDEFINED;
    if size<0> == '0' then
        index = UInt(Q:S);    // S[0-3]
    else
        if S == '1' then UNDEFINED;
        index = UInt(Q);    // D[0-1]
        scale = '11';

```

```

MemOp memop = if L == '1' then MemOp\_LOAD else MemOp\_STORE;
constant integer datasize = 64 << UInt(Q);
constant integer esize = 8 << UInt(scale);

```

Operation

```

CheckFPAdvSIMDEnabled64();

```

```

bits(64) address;
bits(64) offs;
bits(128) rval;
bits(esize) element;
constant integer ebytes = esize DIV 8;

```

```

AccessDescriptor accdesc = CreateAccDescASIMD(memop, nontemporal, tagch

```

```

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

```

```

offs = Zeros(64);
if replicate then
    // load and replicate to all elements
    for s = 0 to selem-1
        element = Mem[address+offs, ebytes, accdesc];
        // replicate to fill 128- or 64-bit register
        V[t, datasize] = Replicate(element, datasize DIV esize);
        offs = offs + ebytes;
        t = (t + 1) MOD 32;

```

```

else
    // load/store one element per register
    for s = 0 to selem-1
        rval = V[t, 128];
        if memop == MemOp\_LOAD then
            // insert into one lane of 128-bit register
            Elem[rval, index, esize] = Mem[address+offs, ebytes, accdesc];
            V[t, 128] = rval;
        else // memop == MemOp_STORE
            // extract from one lane of 128-bit register
            Mem[address+offs, ebytes, accdesc] = Elem[rval, index, esize];
            offs = offs + ebytes;

```

```

        t = (t + 1) MOD 32;

if wback then
    if m != 31 then
        offs = X[m, 64];
    if n == 31 then
        SP[] = address + offs;
    else
        X[n, 64] = address + offs;

```

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
Pseu](#)

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.