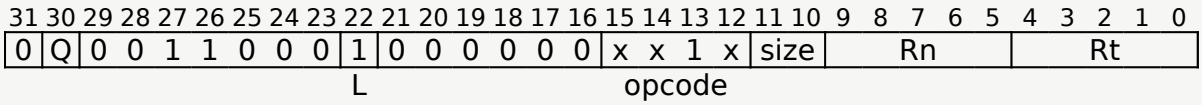
LD1 (multiple structures)

Load multiple single-element structures to one, two, three, or four registers. This instruction loads multiple single-element structures from memory and writes the result to one, two, three, or 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



One register (opcode == 0111)

```
LD1 { <Vt>.<T> }, [<Xn|SP>]
```

Two registers (opcode == 1010)

```
LD1 { <Vt>.<T>, <Vt2>.<T> }, [<Xn|SP>]
```

Three registers (opcode == 0110)

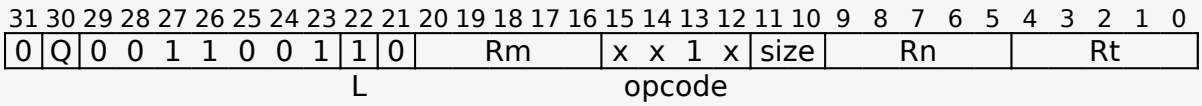
```
LD1 { <Vt>.<T>, <Vt2>.<T>, <Vt3>.<T> }, [<Xn|SP>]
```

Four registers (opcode == 0010)

```
LD1 { <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



One register, immediate offset (Rm == 11111 && opcode == 0111)

```
LD1 { <Vt>.<T> }, [<Xn|SP>], <imm>
```

One register, register offset (Rm != 11111 && opcode == 0111)

```
LD1 { <Vt>.<T> }, [<Xn|SP>], <Xm>
```

Two registers, immediate offset (Rm == 11111 && opcode == 1010)

```
LD1 { <Vt>.<T>, <Vt2>.<T> }, [<Xn|SP>], <imm>
```

Two registers, register offset (Rm != 11111 && opcode == 1010)

```
LD1 { <Vt>.<T>, <Vt2>.<T> }, [<Xn|SP>], <Xm>
```

Three registers, immediate offset (Rm == 11111 && opcode == 0110)

```
LD1 { <Vt>.<T>, <Vt2>.<T>, <Vt3>.<T> }, [<Xn|SP>], <imm>
```

Three registers, register offset (Rm != 11111 && opcode == 0110)

```
LD1 { <Vt>.<T>, <Vt2>.<T>, <Vt3>.<T> }, [<Xn|SP>], <Xm>
```

Four registers, immediate offset (Rm == 11111 && opcode == 0010)

```
LD1 { <Vt>.<T>, <Vt2>.<T>, <Vt3>.<T>, <Vt4>.<T> }, [<Xn|SP>], <imm>
```

Four registers, register offset (Rm != 11111 && opcode == 0010)

```
LD1 { <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>

For the one register, immediate offset variant: is the post-index immediate offset, encoded in “Q”:

Q	<imm>
0	#8
1	#16

For the two registers, immediate offset variant: is the post-index immediate offset, encoded in “Q”:

Q	<imm>
0	#16
1	#32

For the three registers, immediate offset variant: is the post-index immediate offset, encoded in “Q”:

Q	<imm>
0	#24
1	#48

For the four registers, immediate offset variant: is the post-index immediate offset, encoded in “Q”:

Q	<imm>
0	#32
1	#64

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

Shared Decode

```
MemOp memop = if L == '1' then MemOp_LOAD else MemOp_STORE;
constant integer datasize = 64 << UInt(Q);
constant integer esize = 8 << UInt(size);
integer elements = datasize DIV esize;

integer rpt;    // number of iterations
integer selem;    // structure elements

case opcode of
    when '0000' rpt = 1; selem = 4;    // LD/ST4 (4 registers)
    when '0010' rpt = 4; selem = 1;    // LD/ST1 (4 registers)
    when '0100' rpt = 1; selem = 3;    // LD/ST3 (3 registers)
    when '0110' rpt = 3; selem = 1;    // LD/ST1 (3 registers)
    when '0111' rpt = 1; selem = 1;    // LD/ST1 (1 register)
    when '1000' rpt = 1; selem = 2;    // LD/ST2 (2 registers)
    when '1010' rpt = 2; selem = 1;    // LD/ST1 (2 registers)
    otherwise UNDEFINED;

// .1D format only permitted with LD1 & ST1
if size:Q == '110' && selem != 1 then UNDEFINED;
```

Operation

```
CheckFPAdvSIMDEnabled64();

bits(64) address;
bits(64) offs;
bits(datasize) rval;
integer tt;
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);
for r = 0 to rpt-1
    for e = 0 to elements-1
        tt = (t + r) MOD 32;
```

```
        for s = 0 to selem-1
            rval = V[tt, datasize];
            if memop == MemOp_LOAD then
                Elem[rval, e, esize] = Mem[address+offs, ebytes, accdesc
                V[tt, datasize] = rval;
            else // memop == MemOp_STORE
                Mem[address+offs, ebytes, accdesc] = Elem[rval, e, esize
            offs = offs + ebytes;
            tt = (tt + 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.

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