

LDNF1W

Contiguous load non-fault unsigned words to vector (immediate index)

Contiguous load with non-faulting behavior of unsigned words to elements of a vector register from the memory address generated by a 64-bit scalar base and immediate index in the range -8 to 7 which is multiplied by the vector's in-memory size, irrespective of predication, and added to the base address. Inactive elements will not cause a read from Device memory or signal a fault, and are set to zero in the destination vector.

This instruction is illegal when executed in Streaming SVE mode, unless FEAT_SME_FA64 is implemented and enabled.

It has encodings from 2 classes: [32-bit element](#) and [64-bit element](#)

32-bit element

| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|----------|----|----|----|----|----|----|----|----|----|----------|----|------|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| 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 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | imm4 | 1 | 0 | 1 | Pg | Rn | Zt | | | | | | | | | | | | | |
| dtype<0> | | | | | | | | | | dtype<0> | | | | | | | | | | | | | | | | | | | | | |

dtype<0>
dtype<0>

LDNF1W { **<Zt>.S** }, **<Pg>/Z**, [**<Xn|SP>**{, **#<imm>**, **MUL VL**}]

```
if !HaveSVE() then UNDEFINED;
integer t = UInt(Zt);
integer n = UInt(Rn);
integer g = UInt(Pg);
constant integer esize = 32;
constant integer msize = 32;
boolean unsigned = TRUE;
integer offset = SInt(imm4);
```

64-bit element

| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|----------|----|----|----|----|----|----|----|----|----|----------|----|------|----|----|----|----|----|----|----|----|----|---|----|---|---|---|---|---|---|----|---|
| 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 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | imm4 | 1 | 0 | 1 | Pg | | | | | | | Rn | | | | | | | Zt | |
| dtype<0> | | | | | | | | | | dtype<0> | | | | | | | | | | | | | | | | | | | | | |

dtype<0>
dtype<0>

LDNF1W { **<Zt>.D** }, **<Pg>/Z**, [**<Xn|SP>**{, **#<imm>**, **MUL VL**}]

```
if !HaveSVE() then UNDEFINED;
integer t = UInt(Zt);
integer n = UInt(Rn);
integer g = UInt(Pg);
constant integer esize = 64;
constant integer msize = 32;
boolean unsigned = TRUE;
integer offset = SInt(imm4);
```

Assembler Symbols

| | |
|---------|---|
| <Zt> | Is the name of the scalable vector register to be transferred, encoded in the "Zt" field. |
| <Pg> | Is the name of the governing scalable predicate register P0-P7, encoded in the "Pg" field. |
| <Xn SP> | Is the 64-bit name of the general-purpose base register or stack pointer, encoded in the "Rn" field. |
| <imm> | Is the optional signed immediate vector offset, in the range -8 to 7, defaulting to 0, encoded in the "imm4" field. |

Operation

```
CheckNonStreamingSVEEnabled();
constant integer VL = CurrentVL;
constant integer PL = VL DIV 8;
constant integer elements = VL DIV esize;
bits(64) base;
bits(PL) mask = P[g, PL];
bits(VL) result;
bits(VL) orig = Z[t, VL];
bits(msize) data;
constant integer mbytes = msize DIV 8;
boolean fault = FALSE;
boolean faulted = FALSE;
boolean unknown = FALSE;
boolean contiguous = TRUE;
boolean tagchecked = n != 31;
AccessDescriptor accdesc = CreateAccDescSVENF(contiguous, tagchecked);

if !AnyActiveElement(mask, esize) then
    if n == 31 && ConstrainUnpredictableBool(Unpredictable_CHECKSPNONEA
        CheckSPAlignment());
else
    if n == 31 then CheckSPAlignment();
    base = if n == 31 then SP[] else X[n, 64];

for e = 0 to elements-1
    if ActivePredicateElement(mask, e, esize) then
        integer eoff = (offset * elements) + e;
        bits(64) addr = base + eoff * mbytes;
        // MemNF[] will return fault=TRUE if access is not performed fo
        (data, fault) = MemNF[addr, mbytes, accdesc];
    else
        (data, fault) = (Zeros(msize), FALSE);

// FFR elements set to FALSE following a suppressed access/fault
faulted = faulted || fault;
if faulted then
    ElemFFR[e, esize] = '0';

// Value becomes CONSTRAINED UNPREDICTABLE after an FFR element is
unknown = unknown || ElemFFR[e, esize] == '0';
if unknown then
    if !fault && ConstrainUnpredictableBool(Unpredictable_SVELDNFDA
```

```

        Elem[result, e, esize] = Extend(data, esize, unsigned);
    elseif ConstrainUnpredictableBool(Unpredictable_SVELDNFZERO) then
        Elem[result, e, esize] = Zeros(esize);
    else // merge
        Elem[result, e, esize] = Elem[orig, e, esize];
    else
        Elem[result, e, esize] = Extend(data, esize, unsigned);
Z[t, VL] = result;

```

[Base
Instructions](#)

[SIMD&FP
Instructions](#)

[SVE
Instructions](#)

[SME
Instructions](#)

[Index by
Encoding](#)

[Sh
Pseudocode](#)

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.