

LDFF1B (vector plus immediate)

Gather load first-fault unsigned bytes to vector (immediate index)

Gather load with first-faulting behavior of unsigned bytes to active elements of a vector register from memory addresses generated by a vector base plus immediate index. The index is in the range 0 to 31. Inactive elements will not cause a read from Device memory or signal faults, 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	0	0	0	1	0	0	0	1	imm5			1	1	1	Pg			Zn			Zt									
msz<1>msz<0>								U ff																							

LDFF1B { **<Zt>.S** }, **<Pg>/Z**, [**<Zn>.S**{, **#<imm>**}]

```
if !HaveSVE() then UNDEFINED;
integer t = UInt(Zt);
integer n = UInt(Zn);
integer g = UInt(Pg);
constant integer esize = 32;
constant integer msize = 8;
boolean unsigned = TRUE;
integer offset = UInt(imm5);
```

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	1	0	0	0	1	0	0	0	1	imm5			1	1	1	Pg		Zn			Zt										
msz<1>msz<0>										U ff																					

LDFF1B { **<Zt>.D** }, **<Pg>/Z**, [**<Zn>.D**{, **#<imm>**}]

```
if !HaveSVE() then UNDEFINED;
integer t = UInt(Zt);
integer n = UInt(Zn);
integer g = UInt(Pg);
constant integer esize = 64;
constant integer msize = 8;
boolean unsigned = TRUE;
integer offset = UInt(imm5);
```

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.
<Zn>	Is the name of the base scalable vector register, encoded in the "Zn" field.
<imm>	Is the optional unsigned immediate byte offset, in the range 0 to 31, defaulting to 0, encoded in the "imm5" field.

Operation

```
CheckNonStreamingSVEEnabled();
constant integer VL = CurrentVL;
constant integer PL = VL DIV 8;
constant integer elements = VL DIV esize;
bits(PL) mask = P[g, PL];
bits(VL) base;
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 = FALSE;
boolean tagchecked = TRUE;
AccessDescriptor accdesc = CreateAccDescSVEFF(contiguous, tagchecked);

if AnyActiveElement(mask, esize) then
    base = Z[n, VL];

assert accdesc.first;

for e = 0 to elements-1
    if ActivePredicateElement(mask, e, esize) then
        bits(64) addr = ZeroExtend(Elem[base, e, esize], 64) + offset *
        if accdesc.first then
            // Mem[] will not return if a fault is detected for the first
            data = Mem[addr, mbytes, accdesc];
            accdesc.first = FALSE;
        else
            // MemNF[] will return fault=TRUE if access is not performed
            (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);
    elsif ConstrainUnpredictableBool(Unpredictable\_SVELDNFZERO) the
        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.