LD1RB

Load and broadcast unsigned byte to vector

Load a single unsigned byte from a memory address generated by a 64-bit scalar base address plus an immediate offset which is in the range 0 to 63. Broadcast the loaded data into all active elements of the destination vector, setting the inactive elements to zero. If all elements are inactive then the instruction will not perform a read from Device memory or cause a data abort.

It has encodings from 4 classes: 8-bit element , 16-bit element , 32-bit element and 64-bit element

8-bit element

31302928272625	24	23	2221201918171615			14	13	1211109876543210		
1 0 0 0 0 1 0	0	0	1	imm6	1	0	0	Pg	Rn	Zt
	dtypeh<1>	dtypeh<0>				dtypel<1>	dtypel<0>			

```
LD1RB { <Zt>.B }, <Pg>/Z, [<Xn|SP>{, #<imm>}]
```

```
if !HaveSVE() && !HaveSME() then UNDEFINED;
integer t = UInt(Zt);
integer n = UInt(Rn);
integer g = UInt(Pg);
constant integer esize = 8;
constant integer msize = 8;
boolean unsigned = TRUE;
integer offset = UInt(imm6);
```

16-bit element

31302928272625	24	23	2221201918171615			14	13	1211109876543210		
1 0 0 0 0 1 0	0	0	1	imm6	1	0	1	Pg	Rn	Zt
	dtypeh<1>	dtypeh<0>				dtypel<1>	dtypel<0>			

```
LD1RB { <Zt>.H }, <Pg>/Z, [<Xn|SP>{, #<imm>}]
```

```
if !HaveSVE() && !HaveSME() then UNDEFINED;
integer t = UInt(Zt);
integer n = UInt(Rn);
integer g = UInt(Pg);
constant integer esize = 16;
constant integer msize = 8;
boolean unsigned = TRUE;
integer offset = UInt(imm6);
```

32-bit element

31302928272625	24	23	2221201918171615			14	13	1211109876543210		
1 0 0 0 0 1 0	0	0	1	imm6	1	1	0	Pg	Rn	Zt

31302928272625	24	23	22	212019181716	15	14	13	121110	98765	43210
	dtypeh<1>	dtypeh<0>				dtypel<1>	dtypel<0>			

LD1RB { <Zt>.S }, <Pg>/Z, [<Xn|SP>{, #<imm>}]

```
if !HaveSVE() && !HaveSME() then UNDEFINED;
integer t = UInt(Zt);
integer n = UInt(Rn);
integer g = UInt(Pg);
constant integer esize = 32;
constant integer msize = 8;
boolean unsigned = TRUE;
integer offset = UInt(imm6);
```

64-bit element

31302928272625	24	23	22	212019181716	15	14	13	121110	98765	43210
1 0 0 0 0 1 0	0	0	1	imm6	1	1	1	Pg	Rn	Zt
	dtypeh<1>	dtypeh<0>				dtypel<1>	dtypel<0>			

LD1RB { <Zt>.D }, <Pg>/Z, [<Xn|SP>{, #<imm>}]

```
if !HaveSVE() && !HaveSME() then UNDEFINED;
integer t = UInt(Zt);
integer n = UInt(Rn);
integer g = UInt(Pg);
constant integer esize = 64;
constant integer msize = 8;
boolean unsigned = TRUE;
integer offset = UInt(imm6);
```

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 unsigned immediate byte offset, in the range 0 to 63, defaulting to 0, encoded in the "imm6" field.

Operation

```
CheckSVEEnabled();
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(msize) data;
```

```
constant integer mbytes = msize DIV 8;
boolean contiguous = TRUE;
boolean nontemporal = FALSE;
boolean tagchecked = n != 31;
AccessDescriptor accdesc = CreateAccDescSVE(MemOp_LOAD, nontemporal, co

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];
    bits(64) addr = base + offset * mbytes;
    data = Mem[addr, mbytes, accdesc];

for e = 0 to elements-1
    if ActivePredicateElement(mask, e, esize) then
        Elem[result, e, esize] = Extend(data, esize, unsigned);
    else
        Elem[result, e, esize] = Zeros(esize);

Z[t, VL] = result;
```

Operational information

If FEAT_SVE2 is implemented or FEAT_SME is implemented, then if PSTATE.DIT is 1, the timing of this instruction is insensitive to the value of the data being loaded or stored when its governing predicate register contains the same value for each execution.

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