<u>ME</u>	Index by	
ictions	Encoding	

Sh

Pseu

SIMD&FP Base **SVE** SN **Instructions** Instru **Instructions** Instructions

## **UUNPKHI, UUNPKLO**

Unsigned unpack and extend half of vector

Unpack elements from the lowest or highest half of the source vector and then zero-extend them to place in elements of twice their size within the destination vector. This instruction is unpredicated.

It has encodings from 2 classes: High half and Low half

# **High half**

```
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 0 0 0 0 1 0 1 size 1
                            1
                              0
                                 0 | 1 | 1 | 0 0 1
                                                1
                                    UH
```

```
UUNPKHI <Zd>.<T>, <Zn>.<Tb>
```

```
if ! <a href="HaveSVE">HaveSME</a>() then UNDEFINED;
if size == '00' then UNDEFINED;
constant integer esize = 8 << UInt(size);</pre>
integer n = UInt(Zn);
integer d = UInt(Zd);
boolean unsigned = TRUE;
boolean hi = TRUE;
```

#### Low half

```
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 0 0 0 0 1 0 1|size|1 1 0 0|1|0|0 0 1 1 1 0
                                                                                   Zd
                                                                    Zn
                                        UH
```

```
UUNPKLO <Zd>. <T>, <Zn>. <Tb>
```

```
if ! <a href="HaveSVE">HaveSME</a>() then UNDEFINED;
if size == '00' then UNDEFINED;
constant integer esize = 8 << UInt(size);</pre>
integer n = UInt(Zn);
integer d = UInt(Zd);
boolean unsigned = TRUE;
boolean hi = FALSE;
```

### **Assembler Symbols**

< 7.d >

Is the name of the destination scalable vector register, encoded in the "Zd" field.

<T>

Is the size specifier, encoded in "size":

size	<t></t>
0.0	RESERVED
01	Н
10	S
11	D

<Zn>

Is the name of the source scalable vector register, encoded in the "Zn" field.

<Tb>

Is the size specifier, encoded in "size":

size	<tb></tb>
0.0	RESERVED
01	В
10	Н
11	S

## **Operation**

```
CheckSVEEnabled();
constant integer VL = CurrentVL;
constant integer PL = VL DIV 8;
constant integer elements = VL DIV esize;
constant integer hsize = esize DIV 2;
bits(VL) operand = Z[n, VL];
bits(VL) result;

for e = 0 to elements-1
    bits(hsize) element = if hi then Elem[operand, e + elements, hsize]
    Elem[result, e, esize] = Extend(element, esize, unsigned);

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

#### **Operational information**

If FEAT\_SVE2 is implemented or FEAT\_SME is implemented, then if PSTATE.DIT is 1:

- The execution time of this instruction is independent of:
  - The values of the data supplied in any of its registers.
  - The values of the NZCV flags.
- The response of this instruction to asynchronous exceptions does not vary based on:
  - The values of the data supplied in any of its registers.
  - The values of the NZCV flags.

<u>Base</u>	SIMD&FP	<u>SVE</u>	<u>SME</u>	<u>Index by</u>
<u>Instructions</u>	<u>Instructions</u>	<u>Instructions</u>	<u>Instructions</u>	Encoding
Instructions	Instructions	Instructions	Instructions	

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  $\hat{A}$  © 2010-2023 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.