WHILEGT (predicate)

While decrementing signed scalar greater than scalar

Generate a predicate that starting from the highest numbered element is true while the decrementing value of the first, signed scalar operand is greater than the second scalar operand and false thereafter down to the lowest numbered element.

The full width of the scalar operands is significant for the purposes of comparison, and the full width first operand is decremented by one for each destination predicate element, irrespective of the predicate result element size. The first general-purpose source register is not itself updated.

The predicate result is placed in the predicate destination register. Sets the first (N), none (Z), !last (C) condition flags based on the predicate result, and the V flag to zero.

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 1 0 0 1 0 1	size	1	Rm	0 0 0	sf	0 0	Rn	1	Pd
					U	lt		eq	

```
    WHILEGT <Pd>.<T>, <R><n>, <R><m>

if !HaveSVE2() && !HaveSME() then UNDEFINED;
constant integer esize = 8 << UInt(size);
constant integer rsize = 32 << UInt(sf);
integer n = UInt(Rn);
integer m = UInt(Rm);
integer d = UInt(Pd);
boolean unsigned = FALSE;
SVECmp op = Cmp_GT;
```

Assembler Symbols

<Pd>   Is the name of the destination scalable predicate register, encoded in the "Pd" field.

<T>
              Is the size specifier, encoded in "size":

size	<T>
00	B
01	H
10	S
11	D

<R>

Is a width specifier, encoded in "sf":

sf	<R>
0	W
1	X

<n>   Is the number [0-30] of the source general-purpose register or the name ZR (31), encoded in the "Rn" field.

<m>   Is the number [0-30] of the source general-purpose register or the name ZR (31), encoded in the "Rm" field.

Operation

```
CheckSVEEnabled();
constant integer VL = CurrentVL;
constant integer PL = VL DIV 8;
constant integer elements = VL DIV esize;
bits(PL) mask = Ones(PL);
bits(rsize) operand1 = X[n, rsize];
bits(rsize) operand2 = X[m, rsize];
bits(PL) result;
boolean last = TRUE;
constant integer psize = esize DIV 8;

for e = elements-1 downto 0
    boolean cond;
    case op of
        when Cmp_GT cond = (Int(operand1, unsigned) >  Int(operand2, un
        when Cmp_GE cond = (Int(operand1, unsigned) >= Int(operand2, un

    last = last && cond;
    bit pbit = if last then '1' else '0';
    Elem[result, e, psize] = ZeroExtend(pbit, psize);
    operand1 = operand1 - 1;

PSTATE.<N,Z,C,V> = PredTest(mask, result, esize);
P[d, PL] = 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.