How to perform a backward, segmented copy-scan
Solution to CS 1 Short Assignment 9 by THC.

Here is one way to perform a backward, segmented copy-scan. We start with segment bits in the list `seg` and data in the list `x`. We assume that the list size n is at most the number p of processors.

index	0	1	2	3	4	5	6	7	8
`seg`	1	0	0	1	0	0	0	1	0
`x`	5	7	8	3	9	4	5	2	6

1. Reverse `x`, calling the result `rev_x`. Do so by setting `rev_x[i]` to `x[n-i-1]` for $i = 0, 1, \ldots, n-1$. Since each position is written separately, this operation takes $O(1)$ parallel time.

index	0	1	2	3	4	5	6	7	8
`seg`	1	0	0	1	0	0	0	1	0
`x`	5	7	8	3	9	4	5	2	6
`rev_x`	6	2	5	4	9	3	8	7	5

2. Compute new segment bits, `back_seg`, so that `back_seg[i]` is 1 if and only if position i *ends* a segment of `x`. Do so by setting `back_seg[i]` to `seg[n-i]` for $i = 1, 2, \ldots, n-1$ and `back_seg[0]` to 1. This operation takes $O(1)$ parallel time.

index	0	1	2	3	4	5	6	7	8
`seg`	1	0	0	1	0	0	0	1	0
`x`	5	7	8	3	9	4	5	2	6
`rev_x`	6	2	5	4	9	3	8	7	5
`back_seg`	1	0	1	0	0	0	1	0	0

3. Perform a segmented (forward) copy-scan on `rev_x` using `back_seg` as the segment bits. Call the result `back_copy`. This operation takes $O(\log n)$ parallel time.

index	0	1	2	3	4	5	6	7	8
`seg`	1	0	0	1	0	0	0	1	0
`x`	5	7	8	3	9	4	5	2	6
`rev_x`	6	2	5	4	9	3	8	7	5
`back_seg`	1	0	1	0	0	0	1	0	0
`back_copy`	6	6	5	5	5	5	8	8	8

4. The `result` list is just `back_copy`, reversed. Set `result[i]` to `back_copy[n-i-1]` for $i = 0, 1, \ldots, n-1$. This operation takes $O(1)$ parallel time.

index	0	1	2	3	4	5	6	7	8
`seg`	1	0	0	1	0	0	0	1	0
`x`	5	7	8	3	9	4	5	2	6
`rev_x`	6	2	5	4	9	3	8	7	5
`back_seg`	1	0	1	0	0	0	1	0	0
`back_copy`	6	6	5	5	5	5	8	8	8
`result`	8	8	8	5	5	5	5	6	6

The total time for steps 1–4 is $O(\log n)$.