1. `bool implies = !a || b;`

2. The four signals divide into 2 classes, sort0 and sort3 choose the minimum or maximum elements, which look like this. sort3 is identical to sort0, with the comparisons changed from $<=$ to $>=$.

```
int sort0 = [
  a <= b && a <= c && a <= d : a,
  b <= a && b <= c && b <= d : b,
  c <= a && c <= b && c <= d : c,
  1                          : d ];
int sort3 = [
  a >= b && a >= c && a >= d : a,
  b >= a && b >= c && b >= d : b,
  c >= a && c >= b && c >= d : c,
  1                          : d ];
```

   sort1 and sort2 are trickier, because first you have to find the minimum (maximum) element, and then choose the minimum (maximum) of the elements remaining when you remove it from consideration. Here is my solution for sort1 (sort2 is identical to sort1 with the comparisons changed).

```
int sort1 = [
  a <= b && a <= c && a <= d :
    [ b <= c && b <= d : b,
      c <= b && c <= d : c,
      1                : d ],
  b <= a && b <= c && b <= d :
    [ a <= c && a <= d : a,
      c <= a && c <= d : c,
      1                : d ],
  c <= a && c <= b && c <= d :
    [ a <= b && a <= d : a,
      b <= a && b <= d : b,
      1                : d ],
  1                        :
    [ a <= b && a <= c : a,
      b <= a && b <= c : b,
      1                : c ] ];
```

3.
```
irmovl  $-1, %ecx
xorl    %ecx, %eax
```

4.
```
rrmovl  %eax, %ecx
xorl    %ebx, %ecx
andl    %ebx, %eax
xorl    %ecx, %eax
```