Problem 2.71 Solution:

This problem highlights the difference between zero extension and sign extension.

A. The function does not perform any sign extension. For example, if we attempt to extract byte 0 from word `0xFF`, we will get 255, rather than -1.

B. The following code uses the trick shown in Problem 2.23 to isolate a particular range of bits and to perform sign extension at the same time. First, we perform a left shift so that the most significant bit of the desired byte is at bit position 31. Then we right shift by 24, moving the byte into the proper position and performing sign extension at the same time.

code/data/xbyte.c

```
1 int xbyte(packed_t word, int bytenum) {
2     int left = word << ((3-bytenum) << 3);
3     return left >> 24;
4 }
```

code/data/xbyte.c

Problem 2.82 Solution:

These "C puzzle" problems are a great way to motivate students to think about the properties of computer arithmetic from a programmer's perspective. Our standard lecture on computer arithmetic starts by showing a set of C puzzles. We then go over the answers at the end.

A. `(x<y) == (-x>-y)`. No, Let `x` = $TMin_{32}$, `y` = 0.

B. `((x+y)<<4) + y-x == 17*y+15*x`. Yes, from the ring properties of two's complement arithmetic.

C. `~x+~y+1 == ~(x+y)`. Yes, $\sim x + \sim y + 1 = (-x-1)+(-y-1)+1 = -(x+y)-1 = \sim(x+y)$.

D. `(ux-uy) == -(unsigned)(y-x)`. Yes. Due to the isomorphism between two's complement and unsigned arithmetic.

E. `((x >> 2) << 2) <= x`. Yes. Right shift rounds toward minus infinity.