Kaela Polnitz

Homework #1

2.71

We are choosing to only look at the 8 bits involving 0xFF (everything else filled with zeros). As a result, if you use the & operand with a negative number (bits starting with 1), you will always get a 0 to begin your pattern of bits (it will always be positive, the signs will not carry over with these operations).

Int xbyte (packed\_t word, int bytenum)

{

return (word >> ((3 - bytenum) << 3)) >> 24;

}

2.82

A. No. A counter example would be X = Tmin and Y = 2.

B. Yes. This is equivalent to 16\*(x+y) + y – x which is equivalent (after distributive property) to 17\*y + 15\*x

C. Yes. If we look at the table in the book, we notice that ~a = -a – 1 (generally):

~x + ~y + 1 = ~(x+y)

= (-x – 1) + (-y – 1) + 1

= -(x+y) – 1

Which is equivalent to if we did ~a generally with (x+y).

D. Yes. The bits are not going to change because the comparison is not dependent on whether or not it is signed or unsigned.

E. Yes. When we shift it back we lose precision, so it could be the original value or something less if it had extra bits that were lost.