Tejas Bhat

UID: 705199445

Discussion 1C

TA: Zhang

Homework 1

**2.71**

A) The given code does not perform sign extension and as such it does not work for negative numbers. This is due to the fact that packed\_t is originally defined as unsigned, and as such all the other signed ints will also be implicitly converted to unsigned. This creates an inaccurate result as the return value is an unsigned integer instead of a signed integer, forcing inaccurate flow in the program. Furthermore, the signed bit is not even preserved when doing the right shift on the unsigned int, thus further breaking the functionality of the code.

B)

int xbyte(packed\_t word, int bytenum)

{

int shiftDistance = (3 - bytenum) << 3;

int newByte = word << shiftDistance;

return newByte >> 24;

}

**2.82**

A) No, this expression does not always yield 1. If x holds the value of INT MIN, then negating x would give a positive value so large that it would exceed INT MAX and wrap around again to become a negative number. Thus, if y was initially any negative number except INT MIN, the left hand side of the expression would be true since x < y, but the right hand side of the expression would evaluate to false since a negative number cannot be greater than a positive number. Due to this special case, this expression does not always evaluate to true.

B) Yes, this expression always yields 1. A left shift by k as in (x + y) << 4 is represented by (x + y) \* 2^4 or 16 (x + y). Expanding out the left hand side of the expression, we get 16x + 16y + y – x or 17y +15x, which is equivalent to the right hand side of the equation. Thus, this expression is always true and always returns 1.

C) Yes, this expression always yields 1. Expanding out the left hand side of the expression using bit arithmetic, we see that ~x + ~y + 1 is simply (-x-1) + (-y-1) + 1 or (-x-y-1). Now, the right hand side of the expression expands out to –(x+y) – 1, which ends up becoming –x-y-1. Thus, it is clear that the left and right hand sides of the expression are always equal, so this expression always returns 1.

D) Yes, this expression always yields 1. This can be seen through expanding out the right side of the expression. By doing this, we see that we have – (unsigned y – unsigned x). This gives 2^32 – (uy – ux), which is actually 2^32 – uy + ux. The left hand side of the expression is the same since ux – uy is simply ux + (2^32 – uy). Thus, both the left and right hand sides of the expression are always equivalent, and as such the expression always returns 1.   
  
E) Yes, this expression always yields 1. Shifting to the right by k as in u >> k is represented by u/(2^k) and shifting to the left by k as in u << k is represented by u \* 2^k. Thus x >> 2 represents x/4 and (x/4) << 2 simply represents x. Since x is always <= x, this expression always returns 1.