Anica Wang (305532153)

Professor Reinman

CS 33

9 April 2021

Homework 1

Problem 2.71

1. The code does not correctly extract the right byte, and it also does not effectively handle signed and unsigned numbers because the code fragment that executes & 0xFF turns all the 1s (if any) into 0s, thus removing signs from the numbers and turning them all into unsigned.
2. int xbyte(packed\_t word, int bytenum)

{

int byte = word << ((3 – bytenum) << 3);

return byte >> 24;

}

Problem 2.82

1. Does not always yield 1. If y is 1 (00…01) and x is TMin (100…0), y is greater than x because x is negative. The right half is false because -x is still TMin (100…0) but y is now -1, which in signed is (11…11) but in unsigned y is now the largest number so the right half evaluates to false while the left is true, making this expression false.
2. Always yields 1. Left shifting by 4 is the same as multiplying by 16, so the left side evaluates to 16x + 16y + y – x which equals 17y + 15x, which is identical to the left side of the expression.
3. Always yields 1. The left hand side of the equation simplifies to -x + -y – 1 because the ~ of a variable is the same as taking the negative of the variable and subtracting 1 and the right hand side of the expression also simplifies to -x + -y – 1 because ~ (x + y) is the same as –(x + y) – 1, which simplifies to -x + -y as well, identical to the left side
4. Always yields 1. The difference between 2 numbers is identical no matter signed or unsigned. Although the values may be represented differently, the difference is still the same number, just represented differently using signed and unsigned.
5. Always yields 1. The first shift removes the 2 least significant bits and fills the left side with 1 because signed and the second shift fills the right side with 0, making the result less than the original number.