Sam Dinkelman

10/17/18

CIS 247

Lab 3 report

Introduction:

The intent of this assignment was to write four of the six functions outline in the Lab to perform bitwise operations to solve puzzles using strict constraints.

Process:

So for this program I started off with the first function outlined in the lab which was to write a function that computes the bitwise AND of two integers using only NOT and OR. To accomplish this I wrote a simple function that takes in two parameters x and y. My next line declares two ints.

Int a, b;

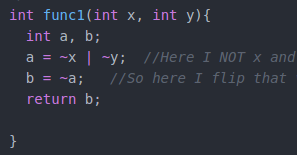
Then I take **a** and set it equal to NOT **x** OR NOT **y**

a = ~x | ~y;

Next I take **b** and set it equal to NOT **a**.

b = ~a;

Then I return **a**. Here is my function in full:



The next Function I completed was Function two outlined in the lab. This function asks us to compute a bitwise XOR using only NOT and AND operators. So I begin by using variables x and y as parameters again. My next line defines three variables **a,** **b,** and **c.**

Int a, b, c;

The next line sets **a** equal to x AND y.

a = x & y;

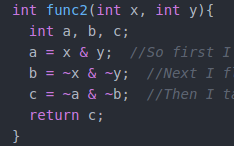
Then after this I use **b** and set that equal to NOT x AND NOT y.

b = ~x & ~y;

After that I take **c** and set that to NOT **a** AND NOT **b**.

c = ~a & ~b;

Then I return c. Here is my function two in full:



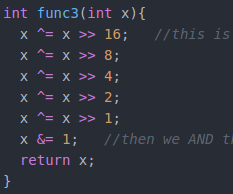
For the third function I had some trouble for a little while. The third function asks us to write a function that returns 1 if any of the odd bits are set to 1 and 0 otherwise. This function takes in x and I didnt define any additional variable for this function. So my first line takes x and shifts it right by 16, then XOR’s the shifted value with the original value.

x ^= x >> 16;

Then for the rest of my lines I run through and shift right by 8, then 4, then 2, then 1 as shown below. My thought process for this was that if we have bits lets say ( a b c d e f g h ) and we shift right by 4 we get ( 0 0 0 0 a b c d) we XOR that with ( a b c d e f g h) we get ( a b c d ea fb gc hd)

Then if we shift right by two we get ( 0 0 a b c d ea fb) and XOR that with (a b c d ea fb gc hd) we get (a b ac bd ace bdf aceg bdfh) so we’re accumlating the bits on the right hand side.

If we shift left once more and XOR we get a value of ( a ab abc abcd abcde abcdef abcdefgh abcdefgh ) so we get all the bits in the right hand side and so when we AND with 1 so that we strip off all the bits except the last one so that the result is 1 if any of the bits were odd.



My last line before the return AND’s the x value by 1 and returns x.

For function 4 I tried lots of different things and eventually with help from some people and some whiteboard work I got my solution. My first line sets a mask to 170, this is the binary value of 1010 1010 so we are going to use this mask to check the bits if they are set to 1.

Int mask = 170;

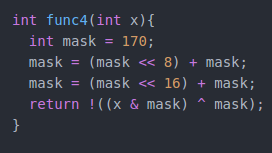
Then my next line sets the mask equal to the value of mask shifted right by 8 so now the mask will be in the second octet instead of the first and then we add the mask back too that so that we get 1010 1010 1010 1010 in total.

mask = (mask << 8) + mask;

Then in the next line I do that same thing but shifted right by 16 and add the mask of 170 so that we get 1010 1010 1010 1010 in the last 2 octets and then add the mask back to it giving us 1010 1010 1010 1010 1010 1010 1010 1010.

mask = (mask << 16) + mask;

From here we do a computation in our return statement that takes the integer value passed by main to this function and AND’s it with the mask this means that for a value opposite the mask it will return all 0’s if the integer x is opposite the mask and the same value of the mask if the integer x is the same as the mask. Then we XOR the result of that with the mask to make sure that if the bits were opposite the mask they flip and if they are the same as the mask they stay the same. When we use the bang character it means NOT so if the x and mask are the same their values would return the same thing so having this NOT there ensures that we return 1 instead of 0 in this case. Here is my function 4 in full:

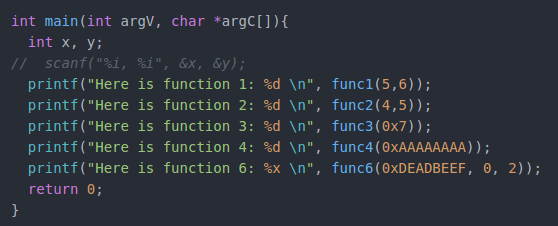


Testing:

To test this program I used all of the values provided by the lab, so for function 1 I tested 5 and 6, my results are below:



It returned 4 which was the correct value for ANDing it.



For function 2 I was testing 4 and 5 and my results returned 1 for the value which is correct with XORing it



For function 3 I was testing it with 0x7 and 0x5 and for 0x7 it would return 1 and for 0x5 it would return 0.









For function 4 I tested the two values 0xFFFFFFFD and 0xAAAAAAAA as outlined by the lab. Here are my results









Result:

So as you can see from the results of my tests I fully implemented the functions as outlined in the assignment. The results we tested the full functionality of these functions since the functions were very basic in their purpose.

Conclusion:

In conclusion I thought this lab provided us with a great opportunity to try out bitwise operators and do various tests to prepare us for the assignment. This lab deepened by understanding of bits and shifting which I only had a very basic understanding of before. I also learned a lot more about AND OR XOR and NOT from this lab which also deepened by understanding of the bits and their various functions.

References and Acknowledgements:

For this lab I did receive assistance from classmates but only once I had already almost completely finished the lab and all of the code was my own work to arrive at the solutions to the functions.