**Description of the program:**

The purpose of this program is to count the instances of the number ‘1’ in a positive binary number. For example, the number 34 is **100010**, so this program would return 2. Also, negative inputs are not allowed, so if the user inputs a negative number, we must output some sort of error message.

**Algorithm Analysis:**

The algorithm I implemented was very simple, but also effective. First, I accept the user input and move that into a global variable. If this input is negative, I output an error and exit. Inside my function I store two words on the stack, a return address, and a local variable (s0). I then check the to see if the function argument is equal to zero. If it does not equal zero, I drop down and use a very simple combination of **andi** and **srl** to check each bit. So, I use **andi** with the immediate value of one, to check the right-most bit against one, if this bit is one then one will be stored, if is zero, then zero will be stored on the stack. I then shift the whole binary sequence to the right (using shift-right logical or **srl**), which shifts the right-most bit. I do this until I have shifted the number until all that is left is zeroes. When it is all zeroes, the function return equals zero, and I jump away to recursively add through what we just stored in the stack. This is returned by the function and printed out.

**Outputs:**



**Proof:** 21 = **1**0**1**0**1**, so output is 3



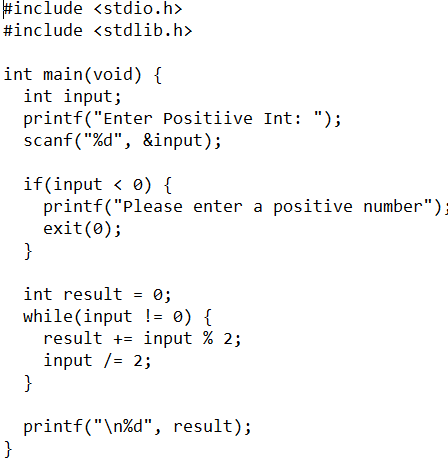
**Proof:** 1018271 = **11111**000**1**00**11**00**11111**, so output is 13



**Proof:** 255 = **11111111**, so output is 8



**C Source Code:**



**MIPS Source Code on Next Pg.**