Ramzey Ghanaim

rghanaim@ucsc.edu

ID: 1395938

Lab 6: Decimal Converter

Due: 5/17/2015

CMPE-12/L: Section 5

Introduction

In this Lab I was able to take my knowledge of assembly and apply it to a bigger task that last week's lab. This new task was to convert a given number, negative or positive into 2's Complement binary. Since I was using assembly, it took a lot of algorithms to implement.

The lab

The first step in the lab was to gather users' input and store it to the integer value of what the user put in. to do this I initialized an integer variable as 0. I took this variable, multiplied it by 10, and added it to the digit the user put in. This is a result of the need to keep track of which position each number is in the decimal number system. The first time a user enters a number we will be multiplying 0 and ten which gives us 0. And thus the int value that the user inputs is just the digit. For the second digit, int is now the first digit the user typed in. For example, if the first digit the user entered was 5 then now int would be 5. If the user then enters 2, the new digit would be 2. Int will be multiplied by 10 and then added to digit. In this case 5 times 10 will equal 50, plus 2 gives us 52. The exact number the user typed in. now int is stored to 52. This was the algorithm I used to take in character input one at a time.

If the user entered a negative number, I had a check to look for the minus sign which turned a flag variable to 1. If 1, the flag will cause the program to invert all the bits, and add 1, to convert to a two's complement number. Basically, one the algorithm to get user's input was complete this algorithm is processed if the user entered a negative value. If not, then we go to masking to find out whether or not the value of each bit will be a 1 or a 0.

In the masking process I first had to load the address of the mask into a register along with a variables called digit, and count to keep track of which bit the program is processing. I then use an LDR command to load the register which contains the address of the mask into another register, with an offset of 0. I then AND the int variable which contains the users' input with the mask and store this into digit. I then invert the bits and add 1 to convert into 2's complement negative. I then check to see if digit is equal to zero by adding the mask to the newly created 2's complement number. If the sum equals zero, then a 1 will be outputted. If not, then a zero will be outputted. This was pretty much all the algorithm for the mask entailed. I then incremented the mask and decremented the counter and made sure count was not negative before repeating the algorithm.

When it came to outputting a 1 or a zero it was really easy to do once I created the masking algorithm to decide which character to input. All I had to do was make sure register 0 was cleared, and add the ASCII code for 1 to print out a 1. I had the same process for printing 0.

Conclusion

In general this lab was more complicated than the previous ones however after breaking down the problem into individual tasks the process of making this program became easier. Also, I was able to increase my understanding of Assembly programming and the functions that I am able to use on the LC-3.