CMPE 310: Systems Design and Programming
Lab Cover Page

Lab #	**2**
Lab Title	**Assembly Language Assignment II (Hamming Distance)**
Name	**Student Name**
Section	**01**
Date Submitted	**MM/DD/YY**

TA / Grader Use Only:

Late Submission Deduction (20% per day late):

Other Deductions:

Final Lab Grade:

Comments to student:

Describe the problem statement
The purpose of assignment two is to write an assembly program that prompts the user for two input strings and computes the hamming distance between the two strings. If the user enters two strings with different lengths, the program should return the hamming distance up to the length of the shorter string. The maximum length of each string can be 255 characters.

Solution Approach
The first part of the code I knew I could accomplish fast was the input from the users. The terminal should prompt the user to enter two strings and read the user input. The next part of the code is calculating the hamming distance between the two strings. In order to accomplish this, two loops are needed. One loop to calculate the total length of the string and we want the smallest of the two strings. The goal of this loop is to loop through the characters in each string. The second loop to rotate the byte to see how many places it differs. This loop needs to run eight times, for each bit in the character and to start again at the next character. Also, counters are needed for the smallest string length and byte. The SI and DI registers need to be initialized to point to the start of string one and string two. A counter is needed to count the difference in the bits. The first bytes in both strings need to be taken into registers and the content needs to be shifted. Then check the condition of the flag register and either increment or don't increment the counter for the hamming distance. Lastly, display the calculated hamming distance on the screen using printf.

Results from the assignment
I started by testing the strings "foo" and "bar" since the value of the hamming distance between those two strings is given in the assignment. When I tested these two values, I got a hamming distance value of eight which is correct. I also tried entering nothing for the user input to make sure my program did not crash. The result for the hamming distance was zero.

Lessons learned
I learned a lot of new instructions in assembly by accomplishing this lab. One of the instructions I used is loops. The JMP instruction can be used for implementing loops. The syntax for loops is generally started with a label. Where, the label is the target label that identifies the target instruction as in the jump instructions. The LOOP instruction assumes that the ECX register contains the loop count. When the loop instruction is executed, the ECX register is decremented and the control jumps to the target label, until the ECX register value, i.e., the counter reaches the value zero. Another instruction I learned from this project is the jump instruction. There are two kinds: conditional and unconditional. The unconditional jump is performed by the JMP instruction. This instruction does not check the flag registers. The conditional jump is performed by a set of jump instructions depending on the condition of the flag registers. I used the compare instruction also. This instruction compares two operands and it is generally used in conditional execution for decision making. Increment and decrement were used in my program. The increment instruction is denoted INC and it is used to increment the

operand by one. The decrement instruction is denoted DEC and is used for decrementing an operand by one. Lastly, I also learned about shifting. This instruction is used to manipulate the binary bits within a register or memory address, so they moved left or right by a specified number of bits and fills in the places with zeros. This instruction is denoted SHL or SHR depending on which way the shift is.

Problems encountered
One problem I encountered was not knowing how to use loops in assembly language. So I just did some reding from an online source to try and understand the concept first. I still didn't quite get it so I decided to go to TA office hours. Which really helped me understand things a lot better. Another error I encountered was, the hamming distance was not printing out. But, with help I found that it was because the line that print it out where not in the exit label. So, when it was jumping to the exit, nothing was being printed it would just end the program. Another problem I was having was with the printf statement. It was not working at first.

How were the problems solved?
My first problem was understanding loops. I overcame this problem by reading about the concepts of loops and getting help from the TA's. After I went to office hours it started making a little bit more sense so I could finish the project. The next problem I encountered was the hamming distance not printing on the terminal. I solved this by moving the print statements into the exit label. By doing this, when the jump to exit is called the program with print the hamming distance and exit the program. The last problem I encountered was the printf statement was causing errors. I fixed it by adding the line "extern printf" to the top of my program because I forgot to do that initially.

Data labels used and what for
I used seven data labels in my assignment. The first one is main which addresses where the program starts. The next label I used is "outer" which denotes the start of my outer loop. This loop is used to loop through each of the characters in the string to find the total length of each and then iterate through the smallest of the two. The next label that is used is "inner" which denotes the start of my inner loop. This loop is designed to run eight times, so it goes through each bit in the character. And to shift the byte to see how many places differ between the two strings. The next label is "test2" which means that the two bits we are comparing are not the same, so the hamming distance gets incremented because the carry flag for the shift of al is one. The "test1" label is used if the two bits are the same. It will jump back to the inner loop and not change anything about the hamming distance because the carry flag for the shift of al is zero. The next label used is "increment" this label increments the hamming distance by one and then jumps back to the inner loop. The last label used is "exit". This label will print the hamming distance and then exit the program.

Description of code
I used multiple variables in my program for each of the messages. Msg1 is "enter string one" and len1 is the length of that message. Msg2 is "Enter string two" and the len2 is the

length of that message. Msg3 is “Hamming distance is” and len3 is the length of that message. Those variables are declared in the data section of my code. The next section of my code is the bss section. This reserves space for the entered strings from the user. Srt1 is the first entered string and length1 is the reserved space for that string. Str2 is for the second entered string from the user and length2 is the reserved length of that string. Each message can be a maximum length of 255 characters. The next section is the text section which holds the actual code. The first part in the main section displays msg1 and msg 2 on the terminal and then allows the user to input there values into str1 and str2. After that, esi and edi registers are used to point to the start of the strings and the edx register is initialized. The next part of the code is all of the loops and data labels which is explained in the section above.

Program Listing

Listing of source code.