Glory Erhabor

CMPE 310 LAB

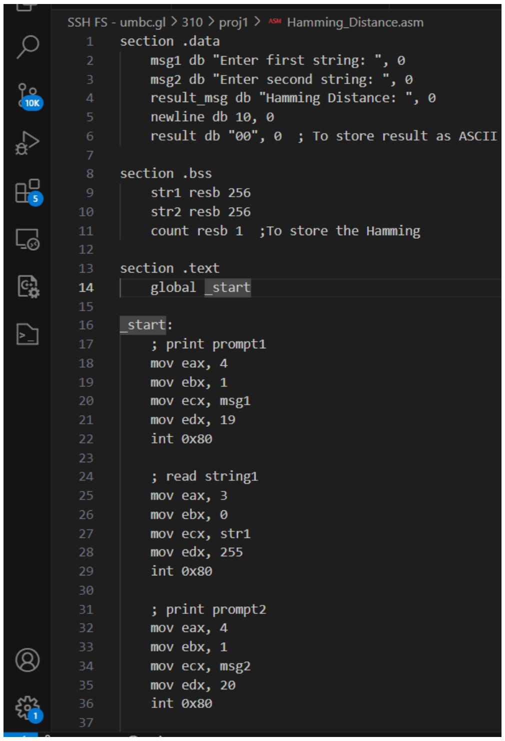
03.06.2025

Professor Kidd

Project 1 Report: Hamming Distance

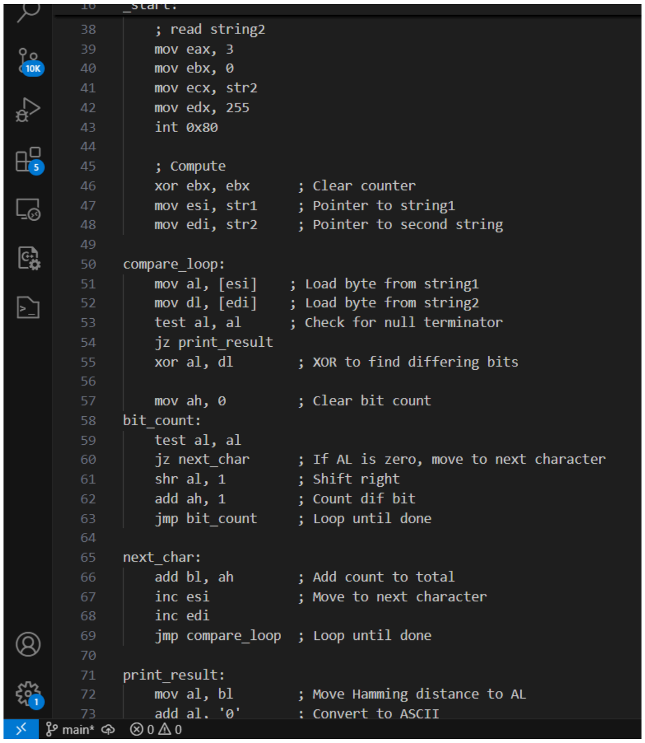
1. When I was creating the program, I began by writing out all the things needed to do. Ask the user to put in a string, determine the binary equivalent, find the distance, then convert to ASCII. The code begins by prompting the user to enter two strings, which are put in 2 different memory locations. It then iterates through both strings character by character, performing a XOR operation on corresponding characters to identify differing bits. The number of differing bits in each character is counted to determine the total Hamming distance. Once the calculation is complete, the result is converted into an ASCII representation and displayed to the user. I used the slides and class examples to develop what should be a working code. When I run the code, it does not display the right ASCII value. Instead it prints incorrect letters and symbols. In order to debug I tried renaming things in case it was confusing the variables. When I tried to use GBD it did work. I asked the TAs as well and they were not sure why it was doing what it was doing as well. This is a learning experience and I will try to do better for the next project.

2. The Code and Outputs

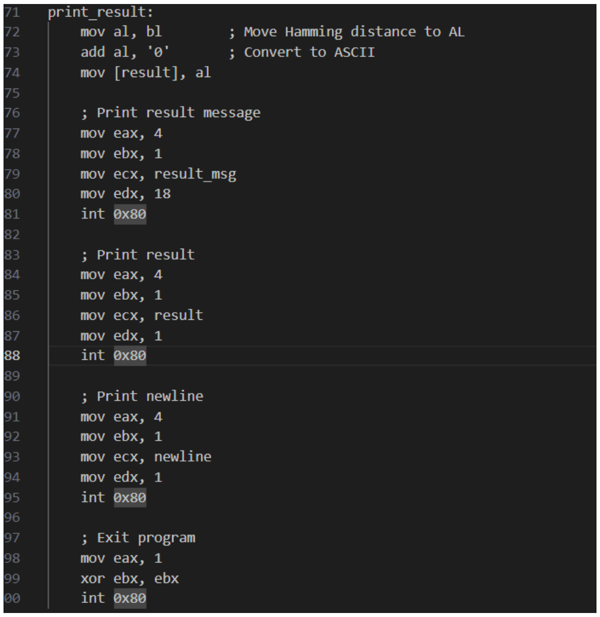


SSH FS - umbc.gl > 310 > proj1 > Hamming_Distance.asm

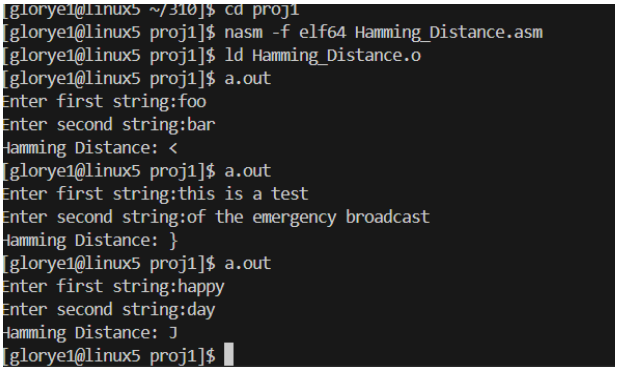
```
section .data
    msg1 db "Enter first string: ", 0
    msg2 db "Enter second string: ", 0
    result_msg db "Hamming Distance: ", 0
    newline db 10, 0
    result db "00", 0  ; To store result as ASCII

section .bss
    str1 resb 256
    str2 resb 256
    count resb 1  ;To store the Hamming

section .text
    global _start

_start:
    ; print prompt1
    mov eax, 4
    mov ebx, 1
    mov ecx, msg1
    mov edx, 19
    int 0x80

    ; read string1
    mov eax, 3
    mov ebx, 0
    mov ecx, str1
    mov edx, 255
    int 0x80

    ; print prompt2
    mov eax, 4
    mov ebx, 1
    mov ecx, msg2
    mov edx, 20
    int 0x80

```



```
_start:
    ; read string2
    mov eax, 3
    mov ebx, 0
    mov ecx, str2
    mov edx, 255
    int 0x80

    ; Compute
    xor ebx, ebx        ; Clear counter
    mov esi, str1       ; Pointer to string1
    mov edi, str2       ; Pointer to second string

compare_loop:
    mov al, [esi]      ; Load byte from string1
    mov dl, [edi]      ; Load byte from string2
    test al, al        ; Check for null terminator
    jz print_result
    xor al, dl          ; XOR to find differing bits

    mov ah, 0           ; Clear bit count
bit_count:
    test al, al
    jz next_char        ; If AL is zero, move to next character
    shr al, 1           ; Shift right
    add ah, 1           ; Count dif bit
    jmp bit_count       ; Loop until done

next_char:
    add bl, ah          ; Add count to total
    inc esi             ; Move to next character
    inc edi
    jmp compare_loop    ; Loop until done

print_result:
    mov al, bl          ; Move Hamming distance to AL
    add al, '0'         ; Convert to ASCII
```



```
print_result:
    mov al, bl          ; Move Hamming distance to AL
    add al, '0'         ; Convert to ASCII
    mov [result], al

    ; Print result message
    mov eax, 4
    mov ebx, 1
    mov ecx, result_msg
    mov edx, 18
    int 0x80

    ; Print result
    mov eax, 4
    mov ebx, 1
    mov ecx, result
    mov edx, 1
    int 0x80

    ; Print newline
    mov eax, 4
    mov ebx, 1
    mov ecx, newline
    mov edx, 1
    int 0x80

    ; Exit program
    mov eax, 1
    xor ebx, ebx
    int 0x80
```



```
[glorye1@linux5 ~/310]$ cd proj1
[glorye1@linux5 proj1]$ nasm -f elf64 Hamming_Distance.asm
[glorye1@linux5 proj1]$ ld Hamming_Distance.o
[glorye1@linux5 proj1]$ a.out
Enter first string:foo
Enter second string:bar
Hamming Distance: <
[glorye1@linux5 proj1]$ a.out
Enter first string:this is a test
Enter second string:of the emergency broadcast
Hamming Distance: }
[glorye1@linux5 proj1]$ a.out
Enter first string:happy
Enter second string:day
Hamming Distance: J
[glorye1@linux5 proj1]$
```