Hamming Distance: Report

Code:

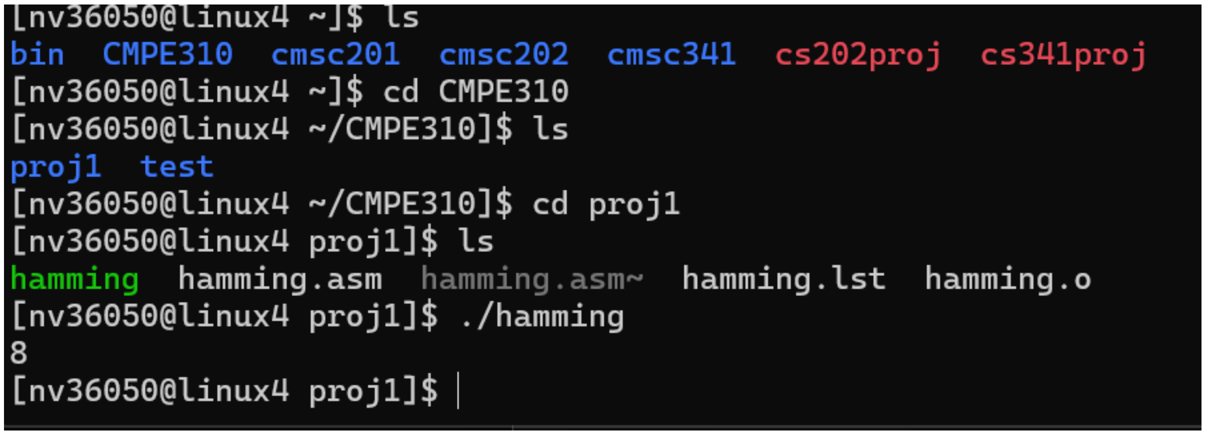
```
section .data
   word1 db 'foo'
   word2 db 'bar'

section .bss
   distance resb 2

section .text
global _start

_start:
   mov esi, word1
   mov edi, word2
   mov ecx, 3
   xor ebx, ebx
wordLoop:
   mov al, [esi]
   xor al, [edi]
bitCounterPrep:
   xor edx, edx
```

```
    mov ebp, 8
bitCountLoop:
    test al, 1
    je nullBit
    add edx, 1
nullBit:
    shr al, 1
    dec ebp
    jne bitCountLoop
    add ebx, edx
    add esi, 1
    add edi, 1
    loop wordLoop
outputPrep:
    mov al, bl
    add al, '0'
    mov ah, 10
    mov [distance], ax

    ;; print
    mov eax, 4
    mov ebx, 1
    mov ecx, distance
    mov edx, 2
    int 0x80


    ;; exit
    mov eax, 1
    xor ebx, ebx
    int 0x80
```

How did I come up with the code:

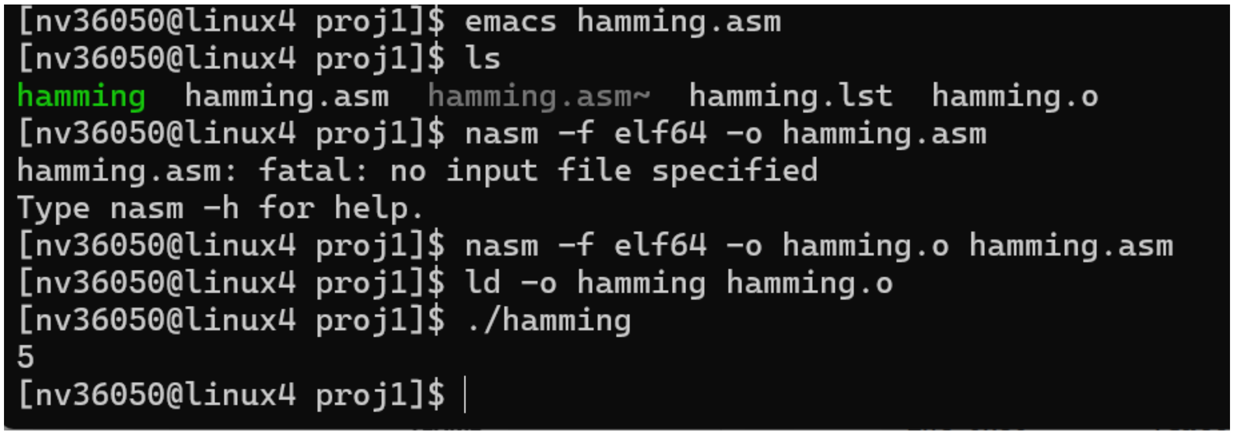
The general idea was to take in 2 strings, “foo”, and “bar”. These were hard coded into the .data section of the file. The functionality could be modified to take in any 2 strings of equal length. The second thing we do is prepare our registers by clearing them with xor. After this we load our characters in binary to our registers. Then we enter our first loop, in this we iterate through to the first character in both strings, stored in the esi and edi registers. With these characters, we xor to find the hamming distance of each character and enter our second loop. In this we iterate through the new xor’d binary to count the 1s. In this case, 1s mean there is a difference in bits between the 2 characters. When we find a 1 in the binary, we increment edx, which stores our output variable, called distance. After we are done iterating through the bits, we enter back into the character loop. This process is repeated until there are no more characters left. Then we convert the distance binary into the ascii representation by adding “0” or 48 to our al register. Then we print out the value, and exit.

Foo Bar output:



```
[nv36050@linux4 ~]$ ls
bin  CMPE310  cmsc201  cmsc202  cmsc341  cs202proj  cs341proj
[nv36050@linux4 ~]$ cd CMPE310
[nv36050@linux4 ~/CMPE310]$ ls
proj1  test
[nv36050@linux4 ~/CMPE310]$ cd proj1
[nv36050@linux4 proj1]$ ls
hamming  hamming.asm  hamming.asm~  hamming.lst  hamming.o
[nv36050@linux4 proj1]$ ./hamming
8
[nv36050@linux4 proj1]$ 
```

Hat Man output:



```
[nv36050@linux4 proj1]$ emacs hamming.asm
[nv36050@linux4 proj1]$ ls
hamming  hamming.asm  hamming.asm~  hamming.lst  hamming.o
[nv36050@linux4 proj1]$ nasm -f elf64 -o hamming.asm
hamming.asm: fatal: no input file specified
Type nasm -h for help.
[nv36050@linux4 proj1]$ nasm -f elf64 -o hamming.o hamming.asm
[nv36050@linux4 proj1]$ ld -o hamming hamming.o
[nv36050@linux4 proj1]$ ./hamming
5
[nv36050@linux4 proj1]$
```