Hamming Distance Report

Initialize:

Set eax, ecx, edx, esi to zero

Move to changechar

Changechars:

Reset ch

Move 1 byte from each string into al and ah, using esi to tell which byte

XOR al and ah to get a byte in al where the 1s are from differences in the two bytes and the zeros are from similarities

Increment esi so next time the character is changed it uses the next character.

Move to loop

Loop:

Shift al right by 1 bit, will update the carry flag with the bit that is shifted off. If the carry flag is 1 move to add, otherwise move to endloop

Add:

Increment edx, move to endloop

Endloop:

Increment cl (total count) and ch (byte counter)

If cl is equal to the defined length in bits, go to finish. If not but ch is equal to 8 (8 bits in a byte, 1 byte to a character), move to changechars. If also not true, move to loop.

Finish:

Add ascii value for 0 to edx and put it into sum. Put 4 in eax, 1 in ebx, sum in ecx, and 1 in edx, to print

Code:

```
section .data
foo db 'this is a test', 0x0d
bar db 'of the emergency broadcast', 0x1a
len db 112
section .bss
sum:resb 4
section .text
global _start
_start:

jmp initialize

initialize:
xor eax, eax
xor ecx, ecx
xor edx, edx
xor esi, esi
jmp changechars
changechars:
xor ch, ch
mov byte al, [foo + esi]
mov byte ah, [bar + esi]
xor al, ah
inc esi
```

```
jmp loop
loop:
shr al, 1
jc add
jmp endloop
add:
inc edx
jmp endloop
endloop:
inc cl
inc ch
cmp cl, [len]
jz finish
cmp ch, 8
jz changechars
jmp loop
finish:

add edx, '0'                ;adds 48, so 8 shows '8', but 38 shows 'V'
mov [sum], edx
mov ecx, sum

mov eax, 4
mov ebx, 1
mov edx, 1
```

int 0x80

mov eax, 1

int 0x80

```
section .data
        foo db 'abc', 0x0d
        bar db 'ABC', 0x1a
        len db 24
section .bss
sum:resb 4
section .text
global _start
_start:

jmp initialize
initialize:
        xor eax, eax
        xor ecx, ecx
        xor edx, edx
        xor esi, esi
        jmp changechars
changechars:
        xor ch, ch
        mov byte al, [foo + esi]
        mov byte ah, [bar + esi]
        xor al, ah
        inc esi
        jmp loop
loop:
        shr al, 1
        jc add
        jmp endloop
add:
        inc edx
        jmp endloop
endloop:
        inc cl
        inc ch
        cmp cl, [len]
        jz finish
        cmp ch, 8
        jz changechars
        jmp loop
finish:
        add edx, '0'            ;adds 48, so 8 shows '8', but 38 shows 'V'
        mov [sum], edx
        mov ecx, sum
        mov eax, 4
        mov ebx, 1
        mov edx, 1

        int 0x80
        mov eax, 1
        int 0x80
```

Outputs:

foo and bar

```
[ry02253@linux4 HammingProj]$ hamming
8[ry02253@linux4 HammingProj]$emacs hamming.asm
```

'this is a test’ and ‘of the emergency broadcast’

Distance is 38, and ‘0’ is 48, so 38+48=86 or a capital V in ascii

```
[ry02253@linux4 HammingProj]$ hamming
V[ry02253@linux4 HammingProj]$cd ..
```

Custom Input

‘abc’ and ‘ABC’

A=0x41, B=0x42,C=0x43

a=0x61, b=0x62, c=0x63

0x4 = 0100, 0x6 = 0110, so distance is 1 for each character, so 3 in total

```
[ry02253@linux4 HammingProj]$ hamming
3[ry02253@linux4 HammingProj]$
```