Hamming Distance Lab

Logan Magaha - PL30737, GitHub Accounts: lgackey, hahayupgit

Concept: Implemented a Hamming Distance algorithm. Pseudocode is as follows.

1. Initiate variables and constants for data storage.
2. Ask user to enter two strings, store strings
3. Convert input strings to binary
4. Compare two strings using Binary XOR, store result.
5. Iterate through the resulting XOR value.
    a. If current bit is 1, check next bit.
    b. If current bit is 0, add 1 to Hamming Distance
6. Convert hamming distance to ASCII
7. Output final values.

Code: Below is a copy-paste of my code.

```
section .data

   line db 0xA ; newline for formatting

   inputMsg db 'Please enter a string:', 0x0A ; for asking user for string
   lenInputMsg equ $-inputMsg

   outputMsg db 'Dist: ', 0x0A ; output msg
   lenOutputMsg equ $-outputMsg

section .bss
   str1 resd 100
   str2 resd 100
   dist resd 100

section .text
  global _start

_start:



   ; get input for first string
```

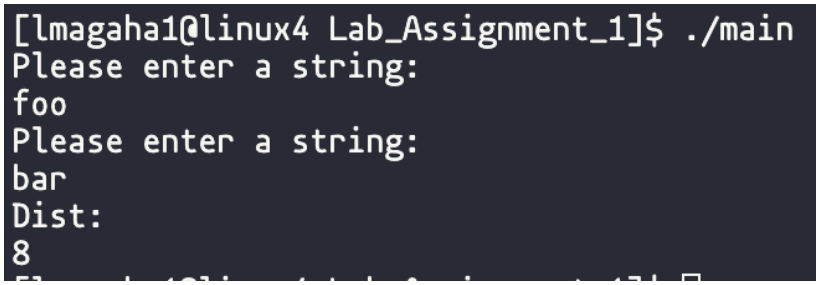
```
mov eax, 4
mov ebx, 1

mov ecx, inputMsg
mov edx, lenInputMsg

int 0x80



; store first string
mov eax, 3
mov ebx, 0

lea ecx, str1
mov edx, 100

int 0x80



; get input for second string
mov eax, 4
mov ebx, 1

mov ecx, inputMsg
mov edx, lenInputMsg

int 0x80



; store second string
mov eax, 3
mov ebx, 0
```

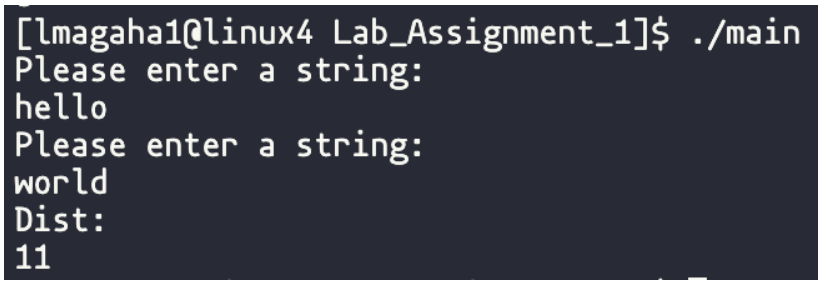
```
    lea ecx, str2
    mov edx, 100

    int 0x80




    ; get binary
    mov eax, [str1]
    mov ebx, [str2]

    ; compare bits, store in eax
    xor eax, ebx

    ; use ecx for hamming dist
    mov ecx, 0

.sum:
    ; check if sig fig bit of eax is 1
    test eax, 1
    ; go to zero if bit is zero
    je .zero ;skip ahead if equal
    ; increase hamming distance otherwise
    inc ecx

.zero:
    ; shift eax right, check next bit
    shr eax, 1
    ; return to sum until all bits checked
    ; because we are only checking eax,
    jne .sum ; loop if not equal

; store data

    ; put hamming dist in eax
    mov eax, ecx
```

```
    ; put ebx in dist buffer
    mov ebx, dist

    ; move to end of buffer
    add ebx, 10

    ; null terminate buffer
    mov byte [ebx], 0

    ; point to last character
    dec ebx

.ascii:
    ; loop
    ; clear remainder
    xor edx, edx
    ; set denominator to 10
    mov ecx, 10
    ; store remainder of eax/10 in edx
    div ecx
    ; convert to ascii
    add dl, '0'
    ; store value in ebx
    mov [ebx], dl
    ; get next character
    dec ebx
    ; check if zero, if not, continue
    test eax, eax
    jne .ascii ;loop if not equal

; printing final things

    ; print outputMsg
    mov eax, 4
    mov ebx, 1
```

```
lea ecx, outputMsg
mov edx, lenOutputMsg
int 0x80



; print dist
mov eax, 4
mov ebx, 1
lea ecx, [dist]
mov edx, 10
int 0x80



; print line
mov eax, 4
mov ebx, 1
lea ecx, [line]
mov edx, 1
int 0x80



; goodbye!
mov eax, 1
xor ebx, ebx
int 0x80
```

Output: Below are screenshots of the requested output.





I have not fully tested every permutation, however I do know that for some inputs, such as “testa” and “testb”, the hamming distance is reported to be 0, where it should be 2. I have tested the code many times and used debuggers, however I have been unable to find an error, and instead I believe this to be an error with my algorithm itself.