Calculating Hamming Distance Using Assembly Programming

Murali Indukuri

February 21, 2025

0.1 Introduction

In this lab, a program to calculate the hamming distance of two strings using the assembly programming language is presented. It works by using the XOR operator to find bits that differ between the two strings and then sums the differing bits. The support up to a kilobyte of text for each input string and can report up to 9,999 differences.

Hamming distance is a measure of the number of bits that differ between two strings. For instance, the bit-strings 01010101 and 10101010 have a hamming distance of 8. All printable characters are encoded using ASCII codes, which are 7-bit numbers. By definition, the XOR operator outputs a high bit only if the inputs differ; therefore, the operator can be used to compute the hamming distance by isolating differing bits. To compute the sum of all the different bits, the result of the XOR operation can be continuously bit shifted to the left or right. If a bit is set high, it will also set the carry flag when it is shifted left or right. Adding the value of the carry flag to a counter will calculate the number of different bits in a given character for both strings. Therefore, an additional loop must be used to loop over all characters. One of the requirements of this program was to only compare the first n characters of both strings where n is the length of the shortest input string. Therefore, the outer loop will break once it finds a newline character in one of the strings. Then, the value of the counter is converted to decimal using integer division. This final value is stored in a single 32-bit register, limiting it to four characters(9,999 differences). The value is printed to the console. Future work may store the final error value on the stack or use 64-bit registers to increase the number of errors reported.

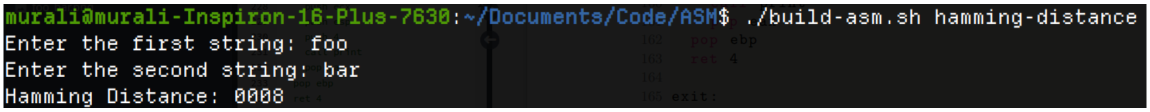
0.2 Full Code

The following shows the full assembly program. Note: the program relies heavily on Linux System Calls, and will not work on any other operating system. Furthermore, it will only work on x86 machines.

```
section .bss
  str1: resb 1025
  str2: resb 1025

section .data
  ;prompt for first string
  str1_prmpt: db "Enter the first string: "
  str1_prmpt_len: equ $-str1_prmpt
  str2_prmpt: db "Enter the second string: "
  str2_prmpt_len: equ $-str2_prmpt
  output_msg: db "Hamming Distance: "
  output_msg_len: equ $-output_msg

section .text
  global _start

_start:
  ; Read input
  ; call prompt print
  push str1_prmpt
push str1_prmpt_len
call print

;get str1 input
mov eax, 3
mov ebx, 0
mov ecx, str1
mov edx, 1025
int 0x80

; call prompt print
push str2_prmpt
push str2_prmpt_len
call print
; get str2 input
mov eax, 3
mov ebx, 0
mov ecx, str2
mov edx, 1025
int 0x80

; Used as counter (DO NOT CHANGE)
xor edx, edx ; use edx for counter
xor edi, edi ; zero out pointer

char_loop:
  xor eax, eax
  mov al, byte [str1+edi]
  ;and eax, 0x000000FF  ;Mask out three bytes

  xor ebx, ebx
  mov bl, byte [str2+edi]
  ;and ebx, 0x000000FF

  ; Exit if EOL
  cmp al, 0x0A
  je report_count
  cmp bl, 0x0A
  je report_count

  xor al, bl ; compare char
  ; if different by 1, eax looks like:
  ; 000000000000001

  ; nested loop
  push edi
  xor edi, edi
  bit_cmp_loop:
    shr al, 1 ; bitshift to carry
    jnc loop_end
  add edx, 1
      loop_end:

      add edi, 1
      cmp edi, 8
      jl bit_cmp_loop
      pop edi

    ; increment count
    add edi, 1
    cmp edi, 1025
    jl char_loop

    report_count:
    push edx
    ; Print output msg
    push output_msg
    push output_msg_len
    call print
    ; Print hamming distance
    ; note that edx is still pushed
    call int_print

    ; Format newline
    push 0x0A ; newline char
    push esp
    push 1
    call print
    call exit


print:
  ; create call frame
  push ebp ; Remember that this is 32bit(4 bytes)
  mov ebp, esp
  ; Function body
  mov eax, 4
  mov ebx, 1
  mov ecx, [ebp+12]
  mov edx, [ebp+8]
  int 0x80
  ; dump stack frame
  pop ebp
  ret 8

int_print:
  push ebp
  mov ebp, esp

  mov eax, [ebp+8] ; get int
  cmp eax, 0
  jne checked_zero
    test:
    mov eax, 0x30303030 ; 0000
    push eax
    push esp
    push 4
    call print
    pop eax
    pop ebp
    ret 4
  checked_zero:

  xor edi, edi
  xor ecx, ecx
  loop_divide:
    ; Divide
    xor edx, edx
    mov ebx, 10
    div ebx
    add edx, 48
    shrd ecx, edx, 8
    add edi, 1
    cmp edi, 4
    jl loop_divide
    ; Output
    ; flip register
    mov ebx, ecx
    xor ecx, ecx
    or ch, bl
    or cl, bh
    ror ecx, 16
    ror ebx, 16
    or ch, bl
    or cl, bh
    ; Finally print
    push ecx
    push esp
    push 4
    call print
    pop eax
  pop ebp
  ret 4

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

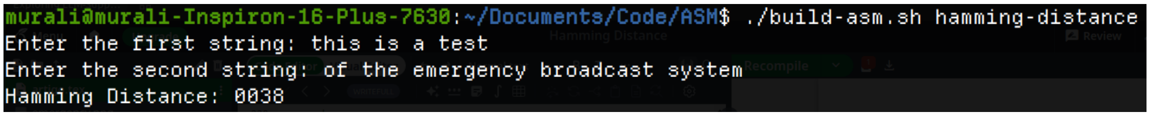
0.3 Sample Output

The following shows the output of the program for selected inputs:



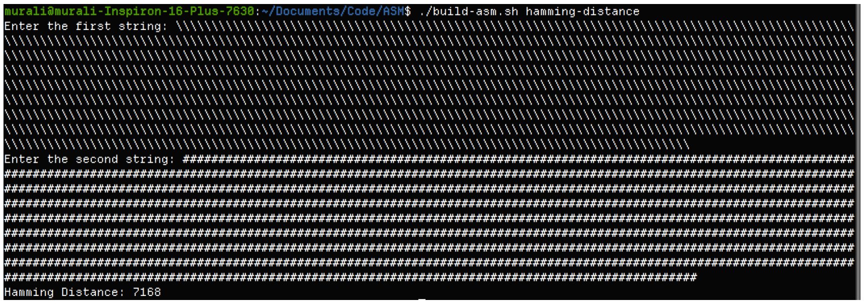
```
murali@murali-Inspiron-16-Plus-7630:~/Documents/Code/ASM$ ./build-asm.sh hamming-distance
Enter the first string: foo
Enter the second string: bar
Hamming Distance: 0008
```

Figure 1: The binary string for foo is 011001100110111101101111. For bar, it is 011000100110000101110010. They differ by 8 bits.



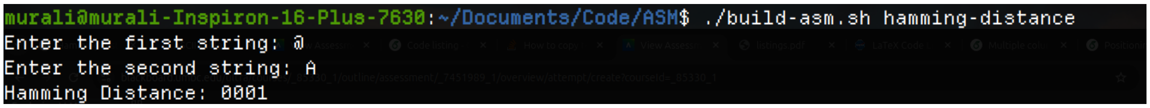
```
murali@murali-Inspiron-16-Plus-7630:~/Documents/Code/ASM$ ./build-asm.sh hamming-distance
Enter the first string: this is a test
Enter the second string: of the emergency broadcast system
Hamming Distance: 0038
```

Figure 2: This shows how the program only considers characters up to the length of the shortest string.



```
murali@murali-Inspiron-16-Plus-7630:~/Documents/Code/ASM$ ./build-asm.sh hamming-distance
Enter the first string: \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
Enter the second string: ##############################################################################################
#######################################################################################################################
#######################################################################################################################
#######################################################################################################################
#######################################################################################################################
#######################################################################################################################
#######################################################################################################################
#######################################################################################################################
#################################################################################################
Hamming Distance: 7168
```

Figure 3: This shows the ability of the program to handle up to a kilobyte(1024 characters) of text. the "#" character differs from the "\" character by 7 bits. As expected, the distance for 1024 characters differing in 7 bits is 7168.



```
murali@murali-Inspiron-16-Plus-7630:~/Documents/Code/ASM$ ./build-asm.sh hamming-distance
Enter the first string: @
Enter the second string: A
Hamming Distance: 0001
```

Figure 4: The "A" character differs from the "@" character by exactly one bit. This was often used for troubleshooting.