```
extern printf
extern gets
extern puts


    section .data
PromptStr1:   db    "Enter first string: ",0
PromptStr2:   db    "Enter second string: ",0
OutputStr:    db    "Hamming Distance = %d",10,0


    section .bss
InStr1:     resb 255                ; Reserve 255 bytes (array) for input
 string 1
InStr2:     resb 255                ; Reserve 255 bytes (array) for input
 string 2

HDistance:  resw 1                  ; Need at least 16 bits for Hamming
 distance counter


    section .text
    global main


main:
    push  dword PromptStr1          ; Push address of first prompt string
     onto stack (argument to printf)
    call  printf                    ; Will output PromptStr1
    push  dword InStr1              ; Push address of InStr1 onto stack
     (argument for gets())
    call  gets                      ; Gets InStr1


    push  dword PromptStr2          ; Push address of second prompt string
     onto stack
    call  printf                    ; Will output PromptStr2
    push  dword InStr2              ; Push address of InStr2 onto stack
    call  gets                      ; Gets Instr2


    mov   esi, 0                    ; Index for outer loop (index for input
     arrays)
StringLoop:
    mov   al, [InStr1 + esi]        ; Move byte from Instr1 at index esi
    mov   bl, [InStr2 + esi]        ; Move byte from Instr2 at index esi

                                    ; Check both bytes loaded to al and bl.
                                     If 0 then at end of string
    cmp   al, 0                     ; End of string?
```

```
    jz    PrintOutput               ; If byte was 0, jump out of outer loop
     to print results
    cmp   bl, 0
    jz    PrintOutput

    xor   al, bl
    mov   ecx, 8                    ; Loop count will be 8 (8 bits per byte)


CheckBit:
    shl   al, 1                     ; Shift reg al left by one and into carry
     flag
    jnc   CheckBitLoop              ; If carry flag not set, skip to end of
     loop

    inc   WORD [HDistance]          ; Increment HDistance, specify word (16
     bit) operation

CheckBitLoop:
    loop  CheckBit                  ; Loops back to CheckBit for next
     iteration
                                    ; Exits when ECX reaches 0
StringLoopBottom:
    inc   esi                       ; Increment index for string loop
    jmp   StringLoop                ; Jump back to outer loop for next byte
     in string

PrintOutput:
    mov   eax, 0                    ; Clear EAX register
    mov   ax, WORD [HDistance]      ; Load lower 16 bits with HDistance value
    push  eax                       ; Push value on to stack
    push  dword OutputStr           ; Push address of format string onto
     stack as 32-bit dword value
    call  printf

    add   esp, 24                   ; Used 24 bytes on stack and returns 24
     to it
                                    ; To point to return address from calling
                                     main

Done:
    ret                             ; Return from main
```