```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;  SYMBOLIC CONSTANT DEFINITIONS   ;;;;;;;;;;;;;;;;;;
null        equu    0x00
MAXARGS     equ     2 ; 1 = program path 2 = 1st arg  3 = 2nd arg etc...
sys_exit    equ     1
sys_read    equ     3
sys_write   equ     4
stdin       equ     0
stdout      equ     1
stderr      equ     3

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;; MACRO DEFINITIONS   ;;;;;;;;;;;;;;;;;;;;;;;;
; print_char macro
; prints one ascii character to the console
%macro print_char 1
    mov eax, 4  ;system call number (sys_write)
    mov ebx, stdout ;file descriptor (stdout)
    mov ecx, %1     ;address of data to print
    mov edx, 1  ;number of bytes to print
    int 0x80    ;do it!
%endmacro

%macro pushRegisters 0
    push eax
    push ebx
    push ecx
    push edx
%endmacro

%macro popRegisters 0
    pop edx
    pop ecx
    pop ebx
    pop eax
%endmacro

; exit0 macro
; exits program with return code 0
%macro exit0 0
    mov ebx, 0
    mov eax, sys_exit
    int 0x80
%endmacro
;;;;;;;;;;;;;;;;;;;;    END MACRO DEFINITIONS   ;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;    DATA SEGMENT    ;;;;;;;;;;;;;;;;;;;;

section .data
var1: db 0xff
var2: db 0xee
nl: db 0x0a, 0x0d
msg_notEQ: db 'The byte values are NOT equal', 0x00
msg_EQ: db  'The byte values ARE equal', 0x00
msg_prompt1: db 'Please enter the first byte: ', 0x00
msg_prompt2: db 'Please enter the second byte: ', 0x00


section .text
  GLOBAL _start
  _start:
    mov edi, msg_prompt1
    call print_string
      ;;get the first byte from user
    mov eax, 3
    mov ebx, 2
    mov ecx, var1
```

```
        mov edx, 1
        int 0x80
        call print_nl

        mov edi, msg_prompt2
        call print_string
          ;;get the second byte from user
        mov eax, 3
        mov ebx, 2
        mov ecx, var2
        mov edx, 1
        int 0x80
        call print_nl


      ; assuming var1 and var2 exist and have some values
        mov al, [var1] ; al = variable 1 value
        cmp al, byte [var2] ; the variables couldn't be compared directly so al was used
      ; the values held in two memory locations cannot be compared in a single instruction
      je var1_eq_var2 ; put sum of variable values into the "sum" variabel
      ; if we are here, var1 != var2
        mov edi, msg_notEQ ; put EQUAL message into edi register
        call print_string ; print EQUAL message
        call print_nl ; print new line
        jmp end_main ; go to end of main program section
      var1_eq_var2:
      ; if we are here, var1 == var2
        mov edi, msg_EQ ; put EQUAL message into edi register
        call print_string ; print EQUAL message
        call print_nl ; print new line
      jmp end_main ; go to end of main program section
      end_main:
      exit0


    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
    ;;;;;;;;;;;;;;;;;;;;    FUNCTIONS/SUBROUTINES   ;;;;;;;;;;;;;;;;;;;;
    ;print_nl
    ;print newline
    ;   returns     - nothing
    print_nl:
        pushRegisters
        mov eax, 4  ;system call number (sys_write) - p75 of Assembly Language Tutorial
        mov ebx, 1  ;file descriptor (stdout)
        mov ecx, nl ;address of data to print
        mov edx, 2  ;number of bytes to print
        int 0x80    ;do it!
        popRegisters
        ret

    ;print_string
    ;   recieves    - address of a C-Style String in register edi
    ;               C-Style means null terminated
    ;   uses        - eax, ebx, ecx, edx
    ;   returns     - nothing
    print_string:
        pushRegisters
        mov ecx, edi
        checknull:
        cmp byte [ecx],null
        jz endstring
            print_char ecx
            inc ecx
            jmp checknull
        endstring:
        popRegisters
            ret
```