section .data

msg1 db 10, "BLOCK A DATA: ", 10

msg1Len equ $ - msg1

msg2 db 10, "BLOCK B DATA [After Transferring] ", 10

msg2Len equ $-msg2

; Define arrays

array db 11h, 59h, 33h, 22h, 44h

darr db 00h, 00h, 00h, 00h, 00h

newline db '', 13, 10

section .bss

counter resb 1

result resb 4

%macro write 2

; A macro to display messages

mov eax, 4

mov ebx, 1

mov ecx, %1

mov edx, %2

int 80h

%endmacro

section .text

global \_start

\_start:

write msg1, msg1Len

mov byte[counter], 05

mov esi, array

next:

mov al, [esi]

push esi

call disp

pop esi

inc esi

dec byte[counter]

jnz next

; Copy the array to destination array

mov byte[counter], 05

mov esi, array

mov edi, darr

next2:

mov al, [esi]

mov [edi], al

inc esi

inc edi

dec byte[counter]

jnz next2

; Display message for block B data

write msg2, msg2Len

; Display the contents of the destination array

mov byte[counter], 05

mov edi, darr

next3:

mov al, [edi]

push edi

call disp

pop edi

inc edi

dec byte[counter]

jnz next3

; Exit the program

mov eax, 1

mov ebx, 0

int 80h

; A subroutine to display the contents of a byte in hexadecimal format

disp:

; Initialize registers

mov bl, al

mov edi, result

mov cx, 02

again:

; Rotate the value of BL by 4 bits to get the next nibble

rol bl, 04

mov al, bl

and al, 0fh

; Convert the nibble to a hexadecimal character

cmp al, 09

jg down

add al, 30h

jmp skip1

down:

add al, 37h

skip1:

; Store the hexadecimal character in the result buffer

mov [edi], al

inc edi

dec cx

jnz again

; Display the result buffer and a newline

write result, 2

write newline, 2

ret

OUTPUT:

