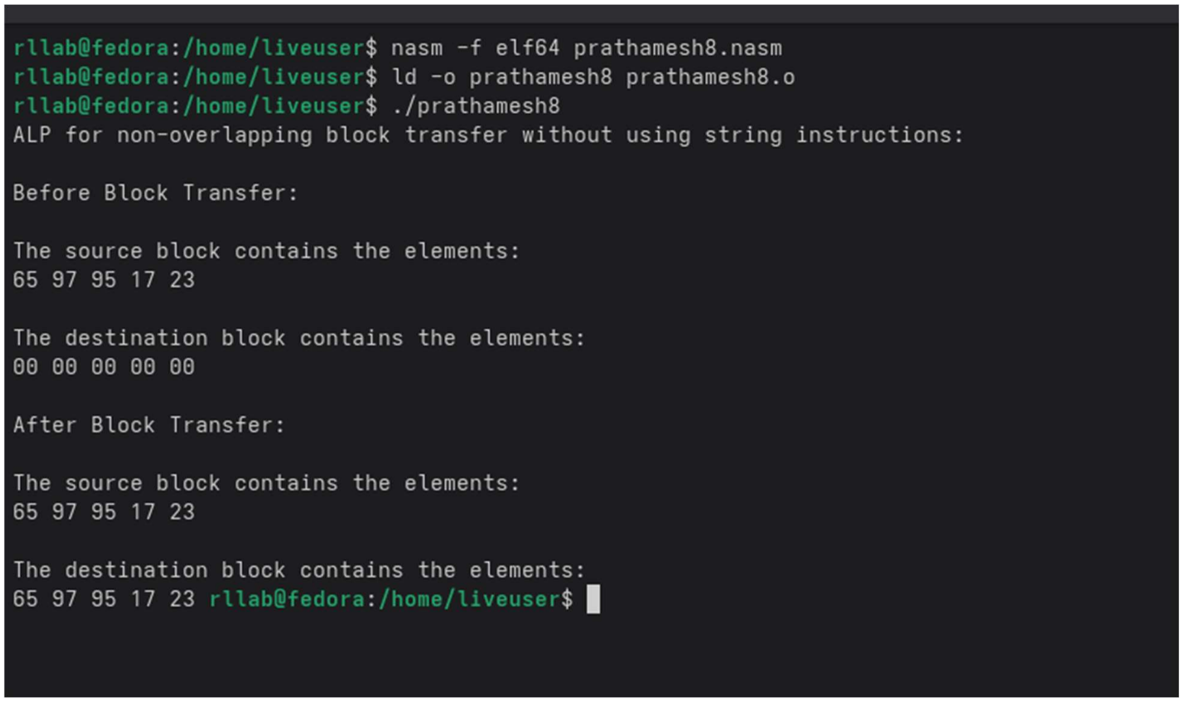
Practical8

```
section .data
sourceBlock db 93h, 94h, 45h, 57h, 13h
count equ 05
msg db "ALP for non-overlapping block transfer without using string
instructions: ", 10
msg_len equ $ - msg
msgSource db 10, "The source block contains the elements: ", 10
msgSource_len equ $ - msgSource
msgDest db 10, 10, "The destination block contains the elements: ", 10
msgDest_len equ $ - msgDest
bef db 10, "Before Block Transfer: ", 10
beflen equ $ - bef
aft db 10, 10, "After Block Transfer: ", 10
aftlen equ $ - aft
space db " "
space_len equ $ - space

section .bss
destBlock resb 5
result resb 4

%macro write 2
    mov rax, 1
    mov rdi, 1
    mov rsi, %1
    mov rdx, %2
    syscall
%endmacro

section .text
global _start
_start:
    write msg, msg_len
    write bef, beflen
    write msgSource, msgSource_len

    ; Display the source block before transfer
    mov rsi, sourceBlock
    call dispBlock

    write msgDest, msgDest_len

    ; Display the destination block before transfer (empty)
    mov rsi, destBlock
    call dispBlock

    ; Perform the block transfer
    mov rsi, sourceBlock
    mov rdi, destBlock
    mov rbp, count
up:
    mov dl, [rsi]      ; Move byte from sourceBlock to dl
    mov [rdi], dl      ; Store byte in destBlock
    inc rsi            ; Move to next byte in sourceBlock
    inc rdi            ; Move to next byte in destBlock
    dec rbp            ; Decrement counter
    jnz up             ; Repeat until count reaches zero

    write aft, aftlen
    write msgSource, msgSource_len

    ; Display the source block after transfer (no change)
    mov rsi, sourceBlock
    call dispBlock

    write msgDest, msgDest_len

    ; Display the destination block after transfer
    mov rsi, destBlock
    call dispBlock

    ; Exit
    mov rax, 60         ; syscall number for exit
    xor rdi, rdi        ; return code 0
    syscall

dispBlock:
    mov rbp, count
next:
    mov al, [rsi]
    push rsi
    call disp
    pop rsi
    inc rsi
    dec rbp
    jnz next
    ret

disp:
    ; Convert byte in AL to hexadecimal and print it
    mov bl, al          ; Copy byte to BL
    mov rdi, result     ; Point rdi to result buffer
    mov cx, 2           ; We need to print 2 hex digits

hex_loop:
    rol bl, 4           ; Rotate byte to left by 4 bits
    mov al, bl          ; Move lower 4 bits to AL
    and al, 0x0F        ;  Mask the upper bits
    cmp al, 9           ; Compare with 9
    jg add_37
    add al, '0'         ; If <= 9, add ASCII value for '0'
    jmp skip1

add_37:
    add al, 'A' - 10    ; If > 9, add ASCII value for 'A' - 10

skip1:
    mov [rdi], al       ; Store ASCII character in result
    inc rdi
    dec cx              ; Decrement count of digits to display
    jnz hex_loop        ; Repeat for second hex digit

    write result, 2     ; Write the two hex digits
    write space, space_len  ; Write space after each number
    ret
```

OUTPUT:



```
rllab@fedora:/home/liveuser$ nasm -f elf64 prathamesh8.nasm
rllab@fedora:/home/liveuser$ ld -o prathamesh8 prathamesh8.o
rllab@fedora:/home/liveuser$ ./prathamesh8
ALP for non-overlapping block transfer without using string instructions:

Before Block Transfer:

The source block contains the elements:
65 97 95 17 23

The destination block contains the elements:
00 00 00 00 00

After Block Transfer:

The source block contains the elements:
65 97 95 17 23

The destination block contains the elements:
65 97 95 17 23 rllab@fedora:/home/liveuser$
```