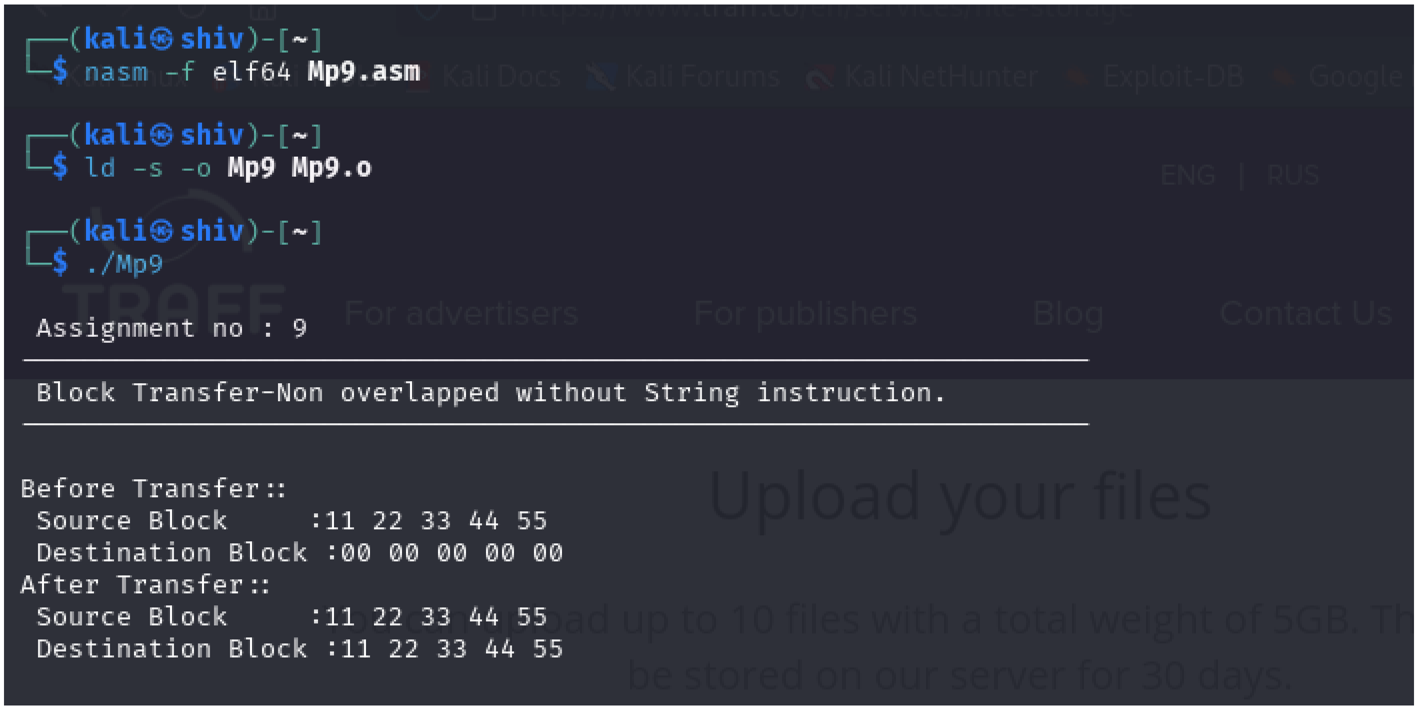
```
section .data
    nline db 10, 10                     ; New line characters
    nline_len equ $ - nline             ; Length of the newline characters

    space db " "                        ; Space for printing
    ano db 10, " Assignment no : 9 "     ; Assignment label
    db                                                         10, "-----------------------------------------------------------------------",
    db 10, " Block Transfer-Non overlapped without String instruction.",
    db 10, "-----------------------------------------------------------------------", 10
    ano_len equ $ - ano                  ; Length of the assignment string

    bmsg db 10, "Before Transfer o"      ; Message before transfer
    bmsg_len equ $ - bmsg                ; Length of 'Before Transfer' message

    amsg db 10, "After Transfer o"       ; Message after transfer
    amsg_len equ $ - amsg                ; Length of 'After Transfer' message

    smsg db 10, " Source Block      :"     ; Source block message
    smsg_len equ $ - smsg                ; Length of 'Source Block' message

    dmsg db 10, " Destination Block :"     ; Destination block message
    dmsg_len equ $ - dmsg                ; Length of 'Destination Block' message

    sblock db 11h, 22h, 33h, 44h, 55h    ; Source block data
    dblock times 5 db 0                  ; Destination block (initialized to 0)

section .bss
    char_ans resB 2                      ; Reserved space for 2 bytes to store characters

%macro Print 2
    MOV RAX, 1                           ; Write syscall
    MOV RDI, 1                           ; File descriptor (stdout)
    MOV RSI, %1                          ; Pointer to the message
    MOV RDX, %2                          ; Message length
    syscall
%endmacro

%macro Read 2
    MOV RAX, 0                           ; Read syscall
    MOV RDI, 0                           ; File descriptor (stdin)
    MOV RSI, %1                          ; Pointer to the buffer
    MOV RDX, %2                          ; Number of bytes to read
    syscall
%endmacro
```

```

%macro Exit 0
    Print nline, nline_len              ; Print a new line
    MOV RAX, G0                         ; Exit syscall
    MOV RDI, 0                          ; Exit code 0
    syscall
%endmacro

section .text
global _start
_start:
    ; Print assignment number and introduction messages
    Print ano, ano_len
    Print bmsg, bmsg_len               ; Block values before transfer
    Print smsg, smsg_len               ; Source Block label

    mov rsi, sblock                    ; Load the address of source block into RSI
    call disp_block                    ; Display source and destination blocks
    Print dmsg, dmsg_len               ; Destination Block label
    mov rsi, dblock                    ; Load the address of destination block into RSI
    call disp_block                    ; Display destination block

    call BT_NO                         ; Perform the block transfer

    ; Print the block values after transfer
    Print amsg, amsg_len               ; After Transfer label
    Print smsg, smsg_len               ; Source Block label
    mov rsi, sblock                    ; Load the address of source block into RSI
    call disp_block                    ; Display source block

    Print dmsg, dmsg_len               ; Destination Block label
    mov rsi, dblock                    ; Load the address of destination block into RSI
    call disp_block                    ; Display destination block

    Exit                                ; Exit the program

;------------------------------------------------------------
BT_NO:
    mov rsi, sblock                    ; Source block pointer
    mov rdi, dblock                    ; Destination block pointer
    mov rcx, 5                          ; Loop counter (5 elements)

back:
    mov al, [rsi]                       ; Load a byte from source block into AL
    mov [rdi], al                       ; Store the byte into destination
```

```
block
    inc rsi                                ; Move to the next byte in source
block
    inc rdi                                ; Move to the next byte in
destination block
    dec rcx                                ; Decrement the counter
    jnz back                               ; Repeat until all elements are
transferred
    RET

;-----------------------------------------------------------------
disp_block:
    mov rbp, 5                             ; Counter for 5 values to display
next_num:
    mov al, [rsi]                          ; Load a byte from source block into
AL
    push rsi                                ; Push RSI onto stack to preserve it
    call Disp_8                             ; Call function to display the byte
as hex
    Print space, 1                          ; Print a space between numbers
    pop rsi                                 ; Restore RSI from stack
    inc rsi                                 ; Move to the next byte in source
block
    dec rbp                                 ; Decrement the counter
    jnz next_num                            ; Repeat until all 5 values are
displayed
    RET

;---------------------------------------------------------------
Disp_8:
    MOV RSI, char_ans + 1                  ; Point to the char_ans buffer
    MOV RCX, 2                             ; Set up the counter (2 hex digits)
    MOV RBX, 16                            ; Set the base to hexadecimal (16)
next_digit:
    XOR RDX, RDX                           ; Clear RDX
    DIV RBX                                ; Divide AL by 16 (hexadecimal)
    CMP DL, 9                              ; Check if the digit is less than or
equal to 9
    JBE add30                              ; If so, add ASCII value for digits
0-9
    ADD DL, 07H                            ; Otherwise, add 7 to make it A-F
add30:
    ADD DL, 30H                            ; Convert the digit to ASCII
    MOV [RSI], DL                          ; Store the digit in char_ans buffer
    DEC RSI                                ; Move to the previous byte in buffer
    DEC RCX                                ; Decrement the counter
    JNZ next_digit                         ; Repeat until both digits are
processed
    Print char_ans, 2                      ; Print the hex representation
    RET
```

Output:



```
┌──(kali㉿shiv)-[~]
└─$ nasm -f elf64 Mp9.asm

┌──(kali㉿shiv)-[~]
└─$ ld -s -o Mp9 Mp9.o

┌──(kali㉿shiv)-[~]
└─$ ./Mp9

 Assignment no : 9
──────────────────────────────────────────────────────────────────
 Block Transfer-Non overlapped without String instruction.
──────────────────────────────────────────────────────────────────

Before Transfer::
 Source Block      :11 22 33 44 55
 Destination Block :00 00 00 00 00
After Transfer::
 Source Block      :11 22 33 44 55
 Destination Block :11 22 33 44 55
```