```
section .data
    sourceBlock db 12h, 45h, 87h, 24h, 97h     ; Source data block (5 elements)
    count equ 05                               ; Number of elements to transfer (5)

    ; Messages for displaying information
    msg db "ALP for non overlapped block transfer using string instructions : ", 10
    msg_len equ $ - msg                        ; Length of msg

    msgSource db 10, "The source block contains the elements : ", 10
    msgSource_len equ $ - msgSource            ; Length of msgSource

    msgDest db 10, 10, "The destination block contains the elements : ", 10
    msgDest_len equ $ - msgDest                ; Length of msgDest

    bef db 10, "Before Block Transfer : ", 10
    beflen equ $ - bef                         ; Length of bef

    aft db 10, 10, "After Block Transfer : ", 10
    aftlen equ $ - aft                         ; Length of aft

section .bss
    destBlock resb 5   ; Reserve 5 bytes for destination block
    result resb 4      ; Reserve 4 bytes for result display

%macro write 2
    mov rax, 1         ; System call for write
    mov rdi, 1         ; File descriptor (stdout)
    mov rsi, %1        ; Source (address of the message)
    mov rdx, %2        ; Length of the message
    syscall            ; Make the system call
%endmacro

section .text
    global _start

_start:
    ; Display the introductory message
    write msg, msg_len

    ; Display "Before Block Transfer" message
    write bef, beflen

    ; Display the source block content
    write msgSource, msgSource_len
    mov rsi, sourceBlock   ; Set rsi to point to sourceBlock
    call dispBlock         ; Call function to display sourceBlock
```

```
47
48      ; Display the destination block content before transfer
49      write msgDest, msgDest_len
50      mov rsi, destBlock      ; Set rsi to point to destBlock
51      call dispBlock          ; Call function to display destBlock
52
53      ; Perform the block transfer using string instruction
54      mov rsi, sourceBlock    ; Set rsi to point to sourceBlock
55      mov rdi, destBlock      ; Set rdi to point to destBlock
5G      mov rcx, count          ; Set rcx to number of elements to transfer
57      cld                     ; Clear direction flag (incrementing addresses)
58      rep movsb               ; Repeat 'movsb' to transfer block of data
59
G0      ; Display "After Block Transfer" message
G1      write aft, aftlen
G2
G3      ; Display the source block content after transfer
G4      write msgSource, msgSource_len
G5      mov rsi, sourceBlock
GG      call dispBlock
G7
G8      ; Display the destination block content after transfer
G9      write msgDest, msgDest_len
70      mov rsi, destBlock
71      call dispBlock
72
73      ; Exit the program
74      mov rax, G0             ; System call number for exit
75      mov rdi, 0              ; Exit status 0
7G      syscall                 ; Make the system call
77
78  ; Function to display block contents
79  dispBlock:
80      mov rbp, count          ; Set rbp to the count of elements
81  next:
82      mov al, [rsi]           ; Load the byte from source (rsi) into al
83      push rsi                ; Save rsi on stack
84      call disp               ; Call the disp function to display the byte
85      pop rsi                 ; Restore rsi from stack
8G      inc rsi                 ; Increment source pointer
87      dec rbp                 ; Decrement count
88      jnz next                ; If count is not zero, repeat loop
89      ret
90
91  ; Function to display a byte in hexadecimal
92  disp:
93      mov bl, al              ; Store the byte in bl
94      mov rdi, result         ; Point rdi to result variable
95      mov cx, 02              ; Load count of rotations (2, for hex display)
9G  up1:
97      rol bl, 04              ; Rotate the number left by 4 bits
```

```
 98      mov al, bl              ; Move lower byte of bl into al
 99      and al, 0fh             ; Mask all but the LSB
100      cmp al, 09h             ; Compare with '9'
101      jg add_37               ; If greater than 9, jump to add 37
102      add al, 30h             ; Otherwise add 30h to get ASCII code for numbers
103      jmp skip1               ; Jump to skip1
104
105  add_37:
10G      add al, 37h             ; If greater than '9', add 37h for hexadecimal
     letters
107
108  skip1:
109      mov [rdi], al           ; Store the ASCII character in result
110      inc rdi                 ; Move to the next byte in result
111      dec cx                  ; Decrement the rotation count
112      jnz up1                 ; Repeat the loop if count is not zero
113
114      ; Write the result (hexadecimal display)
115      write result, 4         ; Write 4 bytes of result (hex digits)
11G      ret
117
```

Output :

```
┌──(kali㉿shiv)-[~]
└─$ gedit Mp8.asm

┌──(kali㉿shiv)-[~]
└─$ nasm -f elf64 Mp8.asm

┌──(kali㉿shiv)-[~]
└─$ ld -s -o Mp8 Mp8.o

┌──(kali㉿shiv)-[~]
└─$ ./Mp8
ALP for non overlapped block transfer using string instructions :

Before Block Transfer :

The source block contains the elements :
1245872497

The destination block contains the elements :
0000000000

After Block Transfer :

The source block contains the elements :
1245872497

The destination block contains the elements :
1245872497
```