# Program No.7-8

; Write X86/64 ALP to perform non-overlapped transfer (with and without string specific instructions). Block containing data can be defined in the data segment.

```
section .data
    sourceBlock db 12h,45h,87h,24h,97h
    count equ 05

    msg db "ALP for non overlapped block transfer 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

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
      mov rsi,sourceBlock
      call dispBlock

      write msgDest , msgDest_len
      mov rsi,destBlock
      call dispBlock

      mov rsi,sourceBlock
      mov rdi,destBlock

      mov rcx, count
      cld
      rep movsb

      write aft , aftlen

      write msgSource , msgSource_len
      mov rsi,sourceBlock
      call dispBlock

      write msgDest , msgDest_len
      mov rsi,destBlock
      call dispBlock

      mov rax,60
      mov rdi,0
      syscall

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

disp:
         mov bl,al ;store number in bl
         mov rdi, result ;point rdi to result variable
```

```
        mov cx,02 ;load count of rotation in cl
up1:
        rol bl,04 ;rotate number left by four bits
        mov al,bl ;move lower byte in dl
        and al,0fh ; get only LSB
        cmp al,09h ;compare with 39h
        jg add_37 ;if grater than 39h skip add 37
        add al,30h
        jmp skip1 ;else add 30
add_37: add al,37h
skip1:  mov [rdi],al ;store ascii code in result variable
        inc rdi ;point to next byte
        dec cx ;decrement the count of digits to display
        jnz up1 ;if not zero jump to repeat

        write result , 4
        ret
```

## Output :