# Practical – 8

**Problem Statement:** Write X86/64 ALP to perform non-overlapped block transfer without string specific instructions. Block containing data can be defined in the data segment.

**Program:**

```
section .data

menumsg db 10,10,'##### Menu for Non-overlapped Block Transfer #####',10

db 10,'1.Block Transfer without using string instructions'

db 10,'2.Block Transfer with using string instructions'

db 10,'3.Exit',10

menumsg_len equ $-menumsg

wrchmsg db 10,10,'Wrong Choice Entered....Please try again!!!',10,10

wrchmsg_len equ $-wrchmsg

blk_bfrmsg db 10,'Block contents before transfer'

blk_bfrmsg_len equ $-blk_bfrmsg

blk_afrmsg db 10,'Block contents after transfer'

blk_afrmsg_len equ $-blk_afrmsg

srcmsg db 10,'Source block contents::'

srcmsg_len equ $-srcmsg

dstmsg db 10,'Destination block contents::'

dstmsg_len equ $-dstmsg

srcblk db 01h,02h,03h,04h,05h

dstblk times 5 db 0

cnt equ 05

spacechar db 20h

lfmsg db 10,10

section .bss
```

```
optionbuff resb 02

dispbuff resb 02

%macro dispmsg 2

mov rax, 01

mov rdi,01

mov rsi,%1

mov rdx,%2

syscall

%endmacro

%macro accept 2

mov rax,00

mov rdi,00

mov rsi,%1

mov rdx,%2

syscall

%endmacro

section .text

global _start

_start:

dispmsg blk_bfrmsg,blk_bfrmsg_len

call showblks

menu: dispmsg menumsg,menumsg_len

accept optionbuff,02

cmp byte [optionbuff],'1'

jne case2

call blkxferwo_proc
```

```
jmp exit1

case2: cmp byte [optionbuff],'2'

jne case3

call blkxferw_proc

jmp exit1

case3: cmp byte [optionbuff],'3'

je exit

dispmsg wrchmsg,wrchmsg_len

jmp menu

exit1:

dispmsg blk_afrmsg,blk_afrmsg_len

call showblks

dispmsg lfmsg,2

exit:

mov rax , 60 ;Exit

mov rdi , 0

syscall

dispblk_proc:

mov rcx,cnt

rdisp:

push rcx

mov bl,[esi] ;Read ASCII value char by char

push rsi

call disp8_proc ;& Display

;Point to next char

dispmsg spacechar,1 ;Display space
```

```
pop rsi
pop rcx
inc esi
loop rdisp ;Decrement count
;Repeat display process till actual count becomes zero
ret
blkxferwo_proc:
mov esi,srcblk
mov edi,dstblk
mov ecx,cnt
blkup1:
mov al,[esi]
mov [edi],al
inc esi
inc edi
loop blkup1
ret
blkxferw_proc:
mov esi,srcblk
mov edi,dstblk
mov ecx,cnt
cld
rep movsb
ret
showblks:
dispmsg srcmsg,srcmsg_len
```

```
mov esi,srcblk
call dispblk_proc
dispmsg dstmsg,dstmsg_len
mov esi,dstblk
call dispblk_proc
ret
disp8_proc:
mov ecx,02
mov edi,dispbuff
dup1:
rol bl,4
mov al,bl
and al,0fh
cmp al,09
jbe dskip
add al,07h
dskip: add al,30h
mov [edi],al
inc edi
loop dup1
dispmsg dispbuff,03
ret
```

**Output:**

```
atharva@atharva:~$ gedit lab8.asm
atharva@atharva:~$ nasm -f elf64 lab8.asm
atharva@atharva:~$ ld -o lab8 lab8.o
```

```
atharva@atharva:~$ ./lab8

Block contents before transfer

Source block contents::01 02 03 04 05

Destination block contents::00 00 00 00 00

##### Menu for Non-overlapped Block Transfer #####

1.Block Transfer without using string instructions

2.Block Transfer with using string instructions

3.Exit

1

Block contents after transfer

Source block contents::01 02 03 04 05

Destination block contents::01 02 03 04 05

atharva@atharva:~$ nasm -f elf64 ass8.asm

atharva@atharva:~$ ld -o ass8 ass8.o

atharva@atharva:~$ ./ass8

Block contents before transfer

Source block contents::01 02 03 04 05

Destination block contents::00 00 00 00 00

##### Menu for Non-overlapped Block Transfer #####

1.Block Transfer without using string instructions

2.Block Transfer with using string instructions

3.Exit

2

Block contents after transfer

Source block contents::01 02 03 04 05

Destination block contents::01 02 03 04 05
```