**Name: Vaishnavi Ranjan**

**Roll no: 205C058**

**Batch: D**

# Assignment no:7

**Title:** 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:** %macro scall 4 ; macro to take input

and output mov rax,%1 mov rdi,%2 mov rsi,%3 mov rdx,%4

syscall

%endmacro

Section .data title: db 0x0A,"------- BLock Transfer -----

------",0x0A

db "Non Overlapped without string", 0x0A t\_len: equ $-title

copy: db 0x0A,0x0A," Copied data", copy\_len: equ $-copy newline: db

0x0A

colon:db " : " colon\_len: equ $-colon cnt\_a: db 05H cnt\_a2:db 05H cnt :db 05H cnt2:db 05H array: db 10H,20H,30H,40H,50H;data to be transferred

;------------- BSS Section -------------------------- Section .bss address: resb 16 val: resb 2 copied: resb 5 choice: resb 2

;------------- MAIN CODE Section -------------------------- Section .text global \_start \_start:

scall 1,1,title,t\_len scall 0,0,choice,2 ;read choice cmp byte[choice],'5' ;if choice==5 then exit je EXIT

;------------- Print Source Array ADDRESS: VALUE --------------- mov byte[cnt\_a],05h mov rsi,array label1: push rsi mov rbx,rsi mov rdi,address call HtoA\_address scall

1,1,newline,1 scall 1,1,address,16 scall 1,1,colon,colon\_len

pop rsi mov bl,byte[rsi] push rsi mov rdi,val call HtoA\_value scall 1,1,val,2 pop rsi inc rsi

dec byte[cnt\_a] jnz label1

;------------- CHOOSE OPTION --------------------------

;compare choice here cmp byte[choice],'1'

JE NONOVERLAPPED

;------Non overlapped copying without string instruction------ NONOVERLAPPED:

;---- Initializaion of starting addresses mov byte[cnt\_a2],5H mov rsi,array mov rdi,array+20H

label2: mov cl,00H mov cl,byte[rsi] mov byte[rdi],cl

inc rsi inc rdi dec byte[cnt\_a2]

jnz label2

jmp OUTPUT

;------OUTPUT of Non-Overlapped ----------------

OUTPUT:

scall 1,1,copy,copy\_len mov byte[cnt\_a],05H mov rsi,array+20H jmp label3

;------Printig ADDRESS:VALUE OF COPIED DATA ---------------- label3: push rsi mov rbx,rsi mov rdi,address call

HtoA\_address scall 1,1,newline,1 scall 1,1,address,16 scall

1,1,colon,colon\_len pop rsi mov bl,byte[rsi]

push rsi mov rdi,val call HtoA\_value

scall 1,1,val,2 pop

rsi

inc rsi

dec byte[cnt\_a] jnz label3

;jmp to start of program jmp \_start EXIT:

mov rax,60 mov

rdi,0 syscall

;------HEX TO ASCII CONVERSION METHOD FOR ADDRESS ----------------

HtoA\_address: ;hex\_no to be converted is in ebx //result is stored in rdi/user defined variable mov byte[cnt2],10H

aup:

rol rbx,04 mov cl,bl and cl,0FH cmp cl,09H jbe ANEXT ADD

cl,07H ANEXT: add cl, 30H mov byte[rdi],cl INC rdi dec byte[cnt2] JNZ aup

ret

;------HEX TO ASCII CONVERSION METHOD FOR VALUE(2 DIGIT) ----------------

HtoA\_value: ;hex\_no to be converted is in ebx //result is stored in rdi/user defined variable mov byte[cnt2],02H aup1: rol bl,04 mov cl,bl and cl,0FH CMP CL,09H jbe ANEXT1

ADD cl,07H

ANEXT1: add cl, 30H mov

byte[rdi],cl INC rdi dec byte[cnt2] JNZ aup1 ret **OUTPUT:**

