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

;perform nonoverlapped block transfer with string instruction

```
section .data

array db 10h,20h,30h,40h,50h

 msg1: db 'Before overlapped :',0xa

len1: equ $-msg1

msg2: db 'After overlapped :',0xa

len2: equ $-msg2

msg3: db ' ',0xa

len3: equ $-msg3

msg4: db ' : '

len4: equ $-msg4


      count db 0

      count1 db 0

      count2 db 0

      count3 db 0

      count4  db 0

section .bss

 addr resb 16

       num1 resb 2
```

```
section .text
global _start
_start:


 mov rax,1
mov rdi,1
mov rsi,msg1
mov rdx,len1
syscall
xor rsi,rsi
mov rsi,array
mov byte[count],05
up:
       mov rbx,rsi
push rsi
 mov rdi,addr
call HtoA1
pop rsi
mov dl,[rsi]
push rsi
 mov rdi,num1
call HtoA2
```

```
pop rsi


inc rsi
dec byte[count]
jnz up


 mov rsi,array
mov rdi,array+5h
mov byte[count3],05h


loop10:
  movsb
 dec byte[count3]
  jnz loop10


 mov rax,1
mov rdi,1
mov rsi,msg2
mov rdx,len2
syscall


      mov rsi,array
      mov byte[count4],0Ah
```

```
        up10:
        mov rbx,rsi
push rsi
        mov rdi,addr
call HtoA1
pop rsi


mov dl,[rsi]
push rsi
        mov rdi,num1
call HtoA2
pop rsi
inc rsi
dec byte[count4]
jnz up10


mov rax,60
mov rdi,0
syscall
HtoA1:
mov byte[count1],16
dup1:
rol rbx,4
```

```
mov al,bl
and al,0fh
cmp al,09
jg p3
add al,30h
jmp p4
p3: add al,37h
p4:mov [rdi],al
inc rdi
 dec byte[count1]
 jnz dup1
 mov rax,1
mov rdi,1
mov rsi,addr
mov rdx,16
syscall
 mov rax,1
mov rdi,1
mov rsi,msg4
mov rdx,len4
syscall
```

```
ret
HtoA2:
mov byte[count2],02
dup2:
rol dl,04
mov al,dl
and al,0fh
cmp al,09h
jg p31
add al,30h
jmp p41

p31: add al,37h
p41:mov [rdi],al
inc rdi
dec byte[count2]
jnz dup2

 mov rax,1
mov rdi,1
mov rsi,num1
mov rdx,02
syscall
```

```
mov rax,1

mov rdi,1

mov rsi,msg3

mov rdx,len3

syscall

ret

;************output*********

;nasm -f elf64 w_over.asm

;ld -o w_over w_over.o

; ./w_over

;Before overlapped :

;0000000000600264 : 10

;0000000000600265 : 20

;0000000000600266 : 30

;0000000000600267 : 40

;0000000000600268 : 50

;After overlapped :

;0000000000600264 : 10

;0000000000600265 : 20

;0000000000600266 : 30

;0000000000600267 : 40

;0000000000600268 : 50
```

;0000000000600269 : 10

;000000000060026A : 20

;000000000060026B : 30

;000000000060026C : 40

;000000000060026D : 50

;perform overlapped block transfer with string instruction

```
section .data
array db 10h,20h,30h,40h,50h
 msg1: db 'Before overlapped :',0xa
len1: equ $-msg1
msg2: db 'After overlapped :',0xa
len2: equ $-msg2
msg3: db ' ',0xa
len3: equ $-msg3
msg4: db ' : '
len4: equ $-msg4


      count db 0
      count1 db 0
      count2 db 0
      count3 db 0
      count4  db 0
      count5  db 0
```

```
section .bss
 addr resb 16
 num1 resb 2
section .text
global _start
_start:
 mov rax,1
mov rdi,1
mov rsi,msg1
mov rdx,len1
syscall
 xor rsi,rsi
mov rsi,array
mov byte[count],05
up:
 mov rbx,rsi
push rsi
 mov rdi,addr
call HtoA1
pop rsi
mov dl,[rsi]
push rsi
 mov rdi,num1
```

```
call HtoA2
pop rsi
inc rsi
dec byte[count]
jnz up
        mov rsi,array
        mov rdi,array+0Ah
       mov byte[count3],05h
       loop10:
        mov dl,00h
        movsb
        dec byte[count3]
        jnz loop10
           xor rsi,rsi
        mov rsi,array+3h
        mov rdi,array+0Ah
       mov byte[count5],05h
 loop11:mov dl,byte[rdi]
        mov byte[rsi],dl
        inc rsi
        inc rdi
        dec byte[count5]
        jnz loop11
```

```
 mov rax,1
mov rdi,1
mov rsi,msg2
mov rdx,len2
syscall
        xor rsi,rsi
        mov rsi,array
        mov byte[count4],08h
        up10:
        mov rbx,rsi
push rsi
 mov rdi,addr
call HtoA1
pop rsi
mov dl,[rsi]
push rsi
 mov rdi,num1
call HtoA2
pop rsi
inc rsi


dec byte[count4]
jnz up10
```

```
 mov rax,60
mov rdi,0
syscall
HtoA1:
mov byte[count1],16
dup1:
rol rbx,4
mov al,bl
and al,0fh
cmp al,09
jg p3
add al,30h
jmp p4
p3: add al,37h
p4:mov [rdi],al
inc rdi
dec byte[count1]
 jnz dup1
mov rax,1
mov rdi,1
mov rsi,addr
```

```
mov rdx,16

syscall

mov rax,1

mov rdi,1

mov rsi,msg4

mov rdx,len4

syscall

ret

HtoA2:

mov byte[count2],02

dup2:

rol dl,04

mov al,dl

and al,0fh

cmp al,09h

jg p31

add al,30h

jmp p41


p31: add al,37h

p41:mov [rdi],al

inc rdi

dec byte[count2]
```

```
        jnz dup2
 mov rax,1
mov rdi,1
mov rsi,num1
mov rdx,02
syscall
mov rax,1
mov rdi,1
mov rsi,msg3
mov rdx,len3
syscall
ret

;******output*******
;nasm -f elf64 wt_over.asm
;ld -o wt_over wt_over.o
; ./wt_over
;Before overlapped :
;000000000060029C : 10
;000000000060029D : 20
;000000000060029E : 30
;000000000060029F : 40
;00000000006002A0 : 50
```

```
;After overlapped :
;000000000060029C : 10
;000000000060029D : 20
;000000000060029E : 30
;000000000060029F : 10
;00000000006002A0 : 20
;00000000006002A1 : 30
;00000000006002A2 : 40
;00000000006002A3 : 50
```

;perform nonoverlapped block transfer without string instruction

```
section .data

array db 10h,20h,30h,40h,50h

      msg1: db 'Before overlapped :',0xa
len1: equ $-msg1

msg2: db 'After overlapped :',0xa
```

```
len2: equ $-msg2

      msg3: db ' ',0xa
len3: equ $-msg3

      msg4: db ' : '
len4: equ $-msg4


      count db 0
      count1 db 0
      count2 db 0
      count3 db 0
      count4  db 0


section .bss
 addr resb 16
       num1 resb 2


section .text
```

```
global _start



_start:



    mov rax,1
mov rdi,1
mov rsi,msg1
mov rdx,len1
syscall

    xor rsi,rsi


mov rsi,array
mov byte[count],05


up:
    mov rbx,rsi
push rsi
    mov rdi,addr
call HtoA1
pop rsi
```

```
mov dl,[rsi]
push rsi
        mov rdi,num1
call HtoA2
pop rsi


inc rsi


dec byte[count]
jnz up



        mov rsi,array
        mov rdi,array+5h
mov byte[count3],05h

       loop10:
        mov dl,00h
        mov dl,byte[rsi]
        mov byte[rdi],dl
        inc rsi
```

```
        inc rdi
        dec byte[count3]
        jnz loop10



        mov rax,1
mov rdi,1
mov rsi,msg2
mov rdx,len2
syscall

        mov rsi,array
        mov byte[count4],0Ah



        up10:
        mov rbx,rsi
push rsi
        mov rdi,addr
call HtoA1
pop rsi
```

```
mov dl,[rsi]
push rsi
    mov rdi,num1
call HtoA2
pop rsi

inc rsi

dec byte[count4]
jnz up10


    mov rax,60
mov rdi,0
syscall

HtoA1:
mov byte[count1],16

dup1:
rol rbx,4
mov al,bl
and al,0fh
```

```
cmp al,09
jg p3
add al,30h
jmp p4
p3: add al,37h
p4:mov [rdi],al
inc rdi
    dec byte[count1]
    jnz dup1


    mov rax,1
mov rdi,1
mov rsi,addr
mov rdx,16
syscall


    mov rax,1
mov rdi,1
mov rsi,msg4
mov rdx,len4
syscall
```

```
ret

HtoA2:
mov byte[count2],02

dup2:

rol dl,04
mov al,dl
and al,0fh
cmp al,09h
jg p31
add al,30h
jmp p41

p31: add al,37h
p41:mov [rdi],al

inc rdi
dec byte[count2]
      jnz dup2
```

```
        mov rax,1
mov rdi,1
mov rsi,num1
mov rdx,02
syscall

        mov rax,1
mov rdi,1
mov rsi,msg3
mov rdx,len3
syscall



ret



;******output*******
```

```
; nasm -f elf64 nonover_string.asm

;ld -o nonover_string nonover_string.o

;./nonover_string

;Before overlapped :

;0000000000600270 : 10

;0000000000600271 : 20

;0000000000600272 : 30

;0000000000600273 : 40

;0000000000600274 : 50

;After overlapped :

;0000000000600270 : 10

;0000000000600271 : 20

;0000000000600272 : 30

;0000000000600273 : 40

;0000000000600274 : 50

;0000000000600275 : 10

;0000000000600276 : 20

;0000000000600277 : 30

;0000000000600278 : 40

;0000000000600279 : 50
```

;perform overlapped block transfer without string instructions

```
section .data

array db 10h,20h,30h,40h,50h

      msg1: db 'Before overlapped :',0xa
len1: equ $-msg1

msg2: db 'After overlapped :',0xa
len2: equ $-msg2

      msg3: db ' ',0xa
len3: equ $-msg3

      msg4: db ' : '
len4: equ $-msg4

      count db 0
      count1 db 0
```

```
        count2 db 0
        count3 db 0
        count4  db 0
        count5  db 0



section .bss
 addr resb 16
         num1 resb 2



section .text
global _start



_start:



        mov rax,1
mov rdi,1
mov rsi,msg1
mov rdx,len1
```

```
syscall

        xor rsi,rsi

mov rsi,array
mov byte[count],05

up:
        mov rbx,rsi
push rsi
        mov rdi,addr
call HtoA1
pop rsi



mov dl,[rsi]
push rsi
        mov rdi,num1
call HtoA2
pop rsi

inc rsi
```

```
dec byte[count]
jnz up


        mov rsi,array
        mov rdi,array+0Ah
mov byte[count3],05h

       loop10:
        mov dl,00h
        mov dl,byte[rsi]
        mov byte[rdi],dl
        inc rsi
        inc rdi
        dec byte[count3]
        jnz loop10

        xor rsi,rsi

        mov rsi,array+3h
        mov rdi,array+0Ah
mov byte[count5],05h
```

```
        loop11:

        mov dl,byte[rdi]
        mov byte[rsi],dl
        inc rsi
        inc rdi
        dec byte[count5]
        jnz loop11




        mov rax,1
mov rdi,1
mov rsi,msg2
mov rdx,len2
syscall

        xor rsi,rsi
        mov rsi,array
        mov byte[count4],08h

        up10:
        mov rbx,rsi
```

```
push rsi
        mov rdi,addr
call HtoA1
pop rsi

mov dl,[rsi]
push rsi
        mov rdi,num1
call HtoA2
pop rsi

inc rsi

dec byte[count4]
jnz up10



        mov rax,60
mov rdi,0
syscall

HtoA1:
mov byte[count1],16
```

```
dup1:
rol rbx,4
mov al,bl
and al,0fh
cmp al,09
jg p3
add al,30h
jmp p4
p3: add al,37h
p4:mov [rdi],al
inc rdi
    dec byte[count1]
    jnz dup1


    mov rax,1
mov rdi,1
mov rsi,addr
mov rdx,16
syscall
```

```
    mov rax,1
mov rdi,1
mov rsi,msg4
mov rdx,len4
syscall


ret

HtoA2:
mov byte[count2],02

dup2:

rol dl,04
mov al,dl
and al,0fh
cmp al,09h
jg p31
add al,30h
jmp p41

p31: add al,37h
```

```
p41:mov [rdi],al


inc rdi
dec byte[count2]
    jnz dup2


    mov rax,1
mov rdi,1
mov rsi,num1
mov rdx,02
syscall


    mov rax,1
mov rdi,1
mov rsi,msg3
mov rdx,len3
syscall



ret
```

```
;********output********
;nasm -f elf64 with_over.asm
; ld -o with_over with_over.o
;./with_over
;Before overlapped :
;00000000006002A4 : 10
;00000000006002A5 : 20
;00000000006002A6 : 30
;00000000006002A7 : 40
;00000000006002A8 : 50
;After overlapped :
;00000000006002A4 : 10
;00000000006002A5 : 20
;00000000006002A6 : 30
;00000000006002A7 : 10
;00000000006002A8 : 20
;00000000006002A9 : 30
;00000000006002AA : 40
;00000000006002AB : 50
```