```
%macro write 2
mov rax , 1
mov rdi , 1
mov rsi , %1
mov rdx , %2
syscall
%endmacro
%macro read 2
mov rax , 0
mov rdi , 0
mov rsi , %1
mov rdx , %2
syscall
%endmacro
section .data

menu db "1. Non-Overlapping",0x0a,"2. Overlapping",0x0A,"5. Exit",0x0A,"Enter your
option: "
menuLen equ $ - menu

m1 db "Enter Count Of numbers: "
l1 equ $-m1
m2 db "Enter Numbers: "
l2 equ $-m2
m3 db "Array 1: "
l3 equ $-m3
m4 db "Array 2: "
l4 equ $-m4
m5 db "Enter Overlapping Position: "
l5 equ $-m5
newLine db 0x0A

section .bss
    optionNo resb 2
    answer resb 20
    array1 resb 300
    array2 resb 300
    count resb 20
    count1 resb 20
    count2 resb 20
    temp resb 20
    posn resb 20

section .text
        global _start
        _start:

    main:
        write menu,menuLen
        read optionNo,2

        cmp byte[optionNo],'5'
        je exit

        call inputarray
```

```
        mov rax,qword[count1]
        mov qword[count],rax
        mov qword[count2],rax
        cmp byte[optionNo],'1'
        je case1
        cmp byte[optionNo],'2'
        je case2
    back:
        mov rax,qword[count2]
        mov qword[count],rax

        call HtoAarray
        jmp main
case1:
    mov rsi,array1
    mov rdi,array2
    loop0:
        mov rax,[rsi]
        mov [rdi],rax
        add rsi,8
        add rdi,8
        dec qword[count]
        jnz loop0
    jmp back

case2:
    write m5,l5
    read temp,17
    call AtoH
    mov qword[posn],rbx
    add qword[count1],rbx
    mov rsi,array1
    mov rdi,array2
    loop1:
        mov rax,[rsi]
        mov [rdi],rax

        add rsi,8
        add rdi,8
        dec qword[count]
        jnz loop1

        mov rax,qword[count2]
        mov qword[count],rax

        mov rsi,array1
        mov rdi,array2

    loop2:
        add rdi,8
        dec qword[posn]
        jnz loop2

    loop3:
        mov rax,[rsi]
```

```
        mov [rdi],rax
        add rsi,8
        add rdi,8
        dec qword[count]
        jnz loop3
    jmp back

exit:
    mov rax,60
    mov rdi,0
    syscall


inputarray:
    write m1,l1
    read temp,17
    call AtoH

    mov qword[count],rbx
    mov qword[count1],rbx

    write m2,l2
    mov rbp,array1

    loop5:
        read temp,17
        call AtoH
        mov qword[rbp],rbx
        add rbp,8
        dec qword[count]
        jnz loop5
    ret

HtoAarray:
    write m3,l3
    mov rbp,array1
    loop6:
        mov rax,[rbp]
        call HtoA
        write newLine,1
        add rbp,8
        dec qword[count]
        jnz loop6
        mov rax,qword[count1]
        mov qword[count],rax
        write m4,l4
        mov rbp,array2
    loop7:
        mov rax,[rbp]
        call HtoA
        write newLine,1
        add rbp,8
        dec qword[count]
        jnz loop7
```

```
ret

AtoH:
    mov rsi,temp
    mov rcx,16
    mov rbx,0
    mov rax,0
    up:
        rol rbx,04
        mov al,[rsi]
        cmp al,39h
        jbe belowNine
        sub al,07h
    belowNine:
        sub al,30h
        add rbx,rax
        inc rsi
        dec rcx
        jnz up
    ret

HtoA:
    mov rsi,answer+15
    mov rcx,16
    up1:
        mov rdx,0
        mov rbx,16
        div rbx
        cmp dl,09h
        jbe belowNine1
        add dl,07h
    belowNine1:
        add dl,30h
        mov [rsi],dl
        dec rsi
        dec rcx
        jnz up1
        write answer,16
    ret
```