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

;write system call to display message ;in the form of print macro

%macro print 2

mov rax,01 ;request to write

mov rdi,01 ;on stdout = screen

mov rsi,%1 ;Ist parameter

mov rdx,%2 ;IInd parameter

syscall

%endmacro

section .data

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

m0 db 10,13,"Non-Overlapping BDT without String instructions"

l0 equ $-m0

m1 db 10,13," Source Block: ",10,13

l1 equ $-m1

m2 db 10," Destinition Block After Transfer: ",10,13

l2 equ $-m2

space db " "

newline db 0xa

section .bss

dstblk resb 05

count resb 01

count1 resb 01

section .text

global \_start

\_start:

print m0,l0 ;display Aim of Program

print m1,l1 ;print srcblk msg

mov rsi,srcblk ;rsi pointing to the base address of srcblk

call disp\_block ;call to procedure named disp\_block

mov rcx,05 ;load counter in counter register

mov rsi,srcblk

mov rdi,dstblk

s1:

mov al,[rsi] ;copy element of srcblk in AL

mov [rdi],al ;paste it in dstblk

inc rsi ;increment pointer in srcblk

inc rdi ;increment pointer in dstblk

loop s1 ;rcx--;

;Compare if rcx =0?; 2 Cases [Y/N]

;If No then jump to the label given in the instruction [label s1]

;If rcx=0 come out of the loop nd goto the next instruction following loop

;instruction

print m2,l2 ;print dstblk message

mov rsi,dstblk ;rsi pointing to the base address of dstblk

call disp\_block ;call to procedure named disp\_block

print newline,1

mov rax, 60

xor rdi, rdi

syscall

;Procedure to display block elements

disp\_block:

mov rbp,05 ;count of array elements

back: mov al,[rsi]

push rsi ;push address of array element on stack

mov bl, al

call disp\_8

print space,1

pop rsi

inc rsi

dec rbp

jnz back

ret

disp\_8:

mov dl, bl

and dl,0f0h

rol dl,04

cmp dl,09h

jbe skip

add dl,07h

skip:

add dl,30h

mov byte[count],dl ;ASCII of higher nibble (Ist digit of n.)

and bl,0fh

cmp bl,09h

jbe skip1

add bl,07h

skip1:

add bl,30h

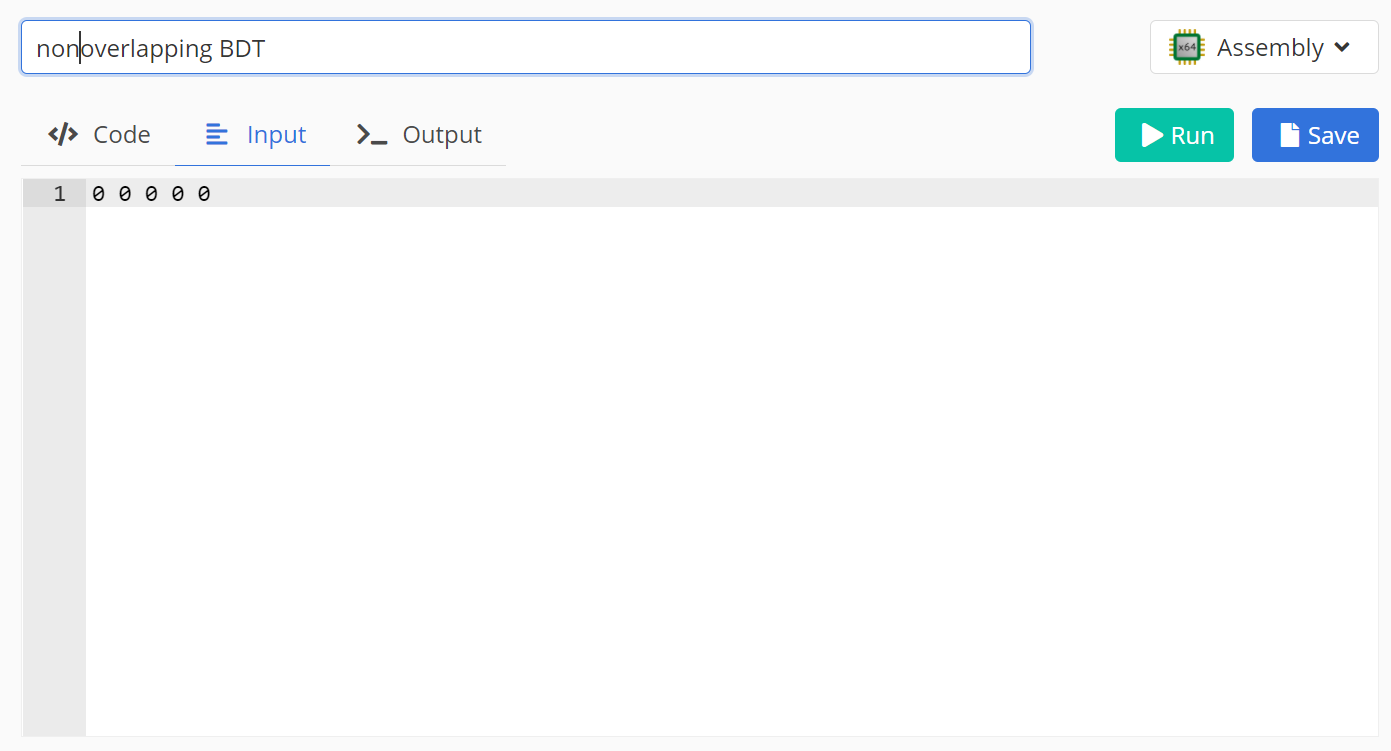
mov byte[count1],bl ;ASCII of lower nibble (IInd digit of n.)

print count,01

print count1,01

ret

INPUT:



OUTPUT:

