8.

; Se da un sir de dublucuvinte. Sa se obtina sirul format din octetii inferiori ai  
;cuvintelor superioare din elementele sirului de dublucuvinte care sunt palindrom in scrierea ;in baza 10.

;Se da sirul de dublucuvinte:

;s **DD** 12345678h, 1A2C3C4Dh, 98FCDC76h

;Sa se obtina sirul

;d **DB** 2Ch, FCh.

bits 32 ; assembling for the 32 bits architecture

; declare the EntryPoint (a label defining the very first instruction of the program)

global start

; declare external functions needed by our program

extern exit ; tell nasm that exit exists even if we won't be defining it

import exit msvcrt.dll ; exit is a function that ends the calling process. It is defined in msvcrt.dll

; msvcrt.dll contains exit, printf and all the other important C-runtime specific functions

; our data is declared here (the variables needed by our program)

segment data use32 class=data

s DD 12345678h, 1A2C3C4Dh, 98FCDC76h

len EQU ($-s)/4

d times len db 0 ; where we store the resulting array

lend DD 0 ; the length of the result

digits times 3 db 0 ; auxiliary array used to store decimal digits. We have at most 3 digits converting byte to decimal

lendigits DD 0 ; length of this auxiliary array

; our code starts here

segment code use32 class=code

start:

; ...

mov ECX, len

mov ESI, s

mov EDI, d

cld

main\_loop:

lodsd ; EAX <- Next dword from s

shr EAX, 16 ; We move the higher part of EAX into AX, so EAX = 0000:AX

mov AH, 0 ; to get only the low part. AL is the byte we work with

push ESI ; We push ESI, ECX and EDI to the stack beacuse

push ECX ; we use these registers to convert from base 16 to base 10

push EDI ; and to check if the number is palindrome

; ------- CONVERTING TO BASE 10 -----------

push EAX ; we keep a copy of the byte we check

mov EDI, digits

second\_loop:

mov BL, 10

div BL; AL = AX / 10, AH = AX % 10

mov DL, AL ; DL = AX / 10. We need this for the next loop

mov AL, AH ; AL = AX mod 10. This is the digit we must be store in memory

stosb; to next value in digits we store current digit, AL

mov AL, DL

mov AH, 0 ; AX = initial number / 10

inc dword [lendigits]

cmp AL, 0

jz stop\_second\_loop ; if AL is 0, we must stop dividing

jmp second\_loop ; if AL is not 0, we get the next digit

stop\_second\_loop:

pop EAX ; AL = byte we got the digits of

; In digits we have our number in decimal. We must check now if it is palindrome

mov ESI, digits

mov EDI, digits

add EDI, [lendigits]

dec EDI ; ESI <- beginning of digits, EDI <- end of digits, digits + lendigits - 1

mov ECX, [lendigits] ; number of digits

shr ECX, 1 ; we divide the number of digits in 2, because we do not need to check all digits for palindromity

cld

palindrom\_loop:

cmpsb; compare pair of digits that should be the same

jne not\_palindrom ; if they are not equal, stop checking

sub EDI, 2 ; EDI became EDI + 1, but we need it to be the previous digit, EDI - 1

loop palindrom\_loop

pop EDI ; EDI <- d

stosb ; we store our current byte (AL) in result array, d

jmp was\_palindrom

not\_palindrom:

pop EDI ; EDI <- d

was\_palindrom:

pop ECX

pop ESI ; ESI <- position in original string

mov dword [lendigits], 0 ; we reset the count of number of digits

loop main\_loop

; exit(0)

push dword 0 ; push the parameter for exit onto the stack

call [exit] ; call exit to terminate the program

