;15. Sa se citeasca de la tastatura doua numere a si b (in baza 10) si sa se calculeze a+b.

; Sa se afiseze rezultatul adunarii in baza 16.

bits 32

global start

extern exit, printf, scanf

import exit msvcrt.dll

import printf msvcrt.dll

import scanf msvcrt.dll

segment data use32 class=data

a dd 0

b dd 0

mesaj db "Rezultatul este", 0

mesaja db "a = ", 0

mesajb db "b = ", 0

format db "%d", 0

formath db "%x", 0

segment code use32 class=code

start:

push dword mesaja

call [printf]

add esp, 4\*1

push dword a

push dword format

call [scanf]

add esp, 4\*2

push dword mesajb

call [printf]

add esp, 4\*1

push dword b

push dword format

call [scanf]

add esp, 4\*2

mov eax, [a]

add eax, [b]

push dword mesaj

push dword eax

push dword formath

call [printf]

add esp, 4\*3

push dword 0

call [exit]

;16. Se da un fisier text. Sa se citeasca continutul fisierului,

; sa se contorizeze numarul de litere 'y' si 'z' si sa se afiseze aceaste valori.

; Numele fisierului text este definit in segmentul de date.

bits 32

global start

extern exit, fopen, fprintf, fclose, printf, fscanf

import exit msvcrt.dll

import fopen msvcrt.dll

import fprintf msvcrt.dll

import fclose msvcrt.dll

import fscanf msvcrt.dll

import printf msvcrt.dll

segment data use32 class=data

nume\_fisier db "tema.txt", 0 ; numele fisierului care va fi deschis

mod\_acces db "r", 0 ; modul de deschidere a fisierului; r - pentru scriere. fisierul trebuie sa existe

descriptor\_fis dd -1 ; variabila in care vom salva descriptorul fisierului - necesar pentru a putea face referire la fisier

; nr\_car\_citite dd 0 ; variabila in care vom salva numarul de caractere citite din fisier in etapa curenta

; lgsir equ 100 ; numarul maxim de elemente citite din fisier intr-o etapa

; sir resb lgsir ; sirul in care se va citi textul din fisier

format db "%d", 0

; z equ 'z'

; y equ 'y'

contz dd 0

conty dd 0

mesajy db "Sunt %d litere y", 0

mesajz db "Sunt %d litere z", 0

caracter db 0

format\_citire db "%c", 0

segment code use32 class=code

start:

; apelam fopen pentru a deschide fisierul

; functia va returna in EAX descriptorul fisierului sau 0 in caz de eroare

; eax = fopen(nume\_fisier, mod\_acces)

push dword mod\_acces

push dword nume\_fisier

call [fopen]

add esp, 4\*2

; verificam daca functia fopen a creat cu succes fisierul (daca EAX != 0)

cmp eax, 0

je final

mov [descriptor\_fis], eax ; salvam valoarea returnata de fopen in variabila descriptor\_fis

repeta:

push dword caracter

push dword format\_citire

push dword [descriptor\_fis]

call [fscanf]

add esp, 4\*3

cmp eax, 1 ; EAX = 1, daca a citit cu SUCCES un caracter

jne afiseaza

; mov bl, [caracter]

; cmp bl, z

cmp byte [caracter], 'z'

jne comparay

inc dword[conty]

comparay:

; cmp bl, y

cmp byte [caracter], 'y'

jne finalcomp

inc dword[contz]

finalcomp:

jmp repeta

afiseaza:

; push dword conty ; aici e OFFSET

push dword [conty] ; e VALOARE

push dword mesajy

call [printf]

add esp, 4\*2

; apelam functia fclose pentru a inchide fisierul

; fclose(descriptor\_fis)

push dword [descriptor\_fis]

call [fclose]

add esp, 4

final:

push dword 0

call [exit]