8.

bits 32 ; assembling for the 32 bits architecture

;Se da un numar natural a (a: dword, definit in segmentul de date).

;Sa se citeasca un numar natural b si sa se calculeze: a + a/b. Sa se afiseze rezultatul operatiei. Valorile vor fi afisate in format decimal (baza 10) cu semn.

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

global start

; declararea functiilor externe folosite de program

extern exit, printf, scanf ; adaugam printf si scanf ca functii externe

import exit msvcrt.dll

import printf msvcrt.dll ; indicam asamblorului ca functia printf se gaseste in libraria msvcrt.dll

import scanf msvcrt.dll ; similar pentru scanf

segment data use32 class=data

a dd 15

b dd 0

format db "%d",0

message db "a+a/b=%d",0

; our code starts here

segment code use32 class=code

start:

push dword b ; push the value where I put the read number

push dword format ; push its format

call [scanf] ; read the number from keyboard

add ESP, 4\*2 ;clear the stack

mov EAX, [a] ; put the value of a in eax

cdq; make eax quadraword

idiv dword[b] ; divide eax quadra to value of b, and becomes again dword

mov EDI, EAX ; edi=eax

add EDI, [a]; edi=eax+a

push dword EDI; push the value to be printed

push dword message; push its format

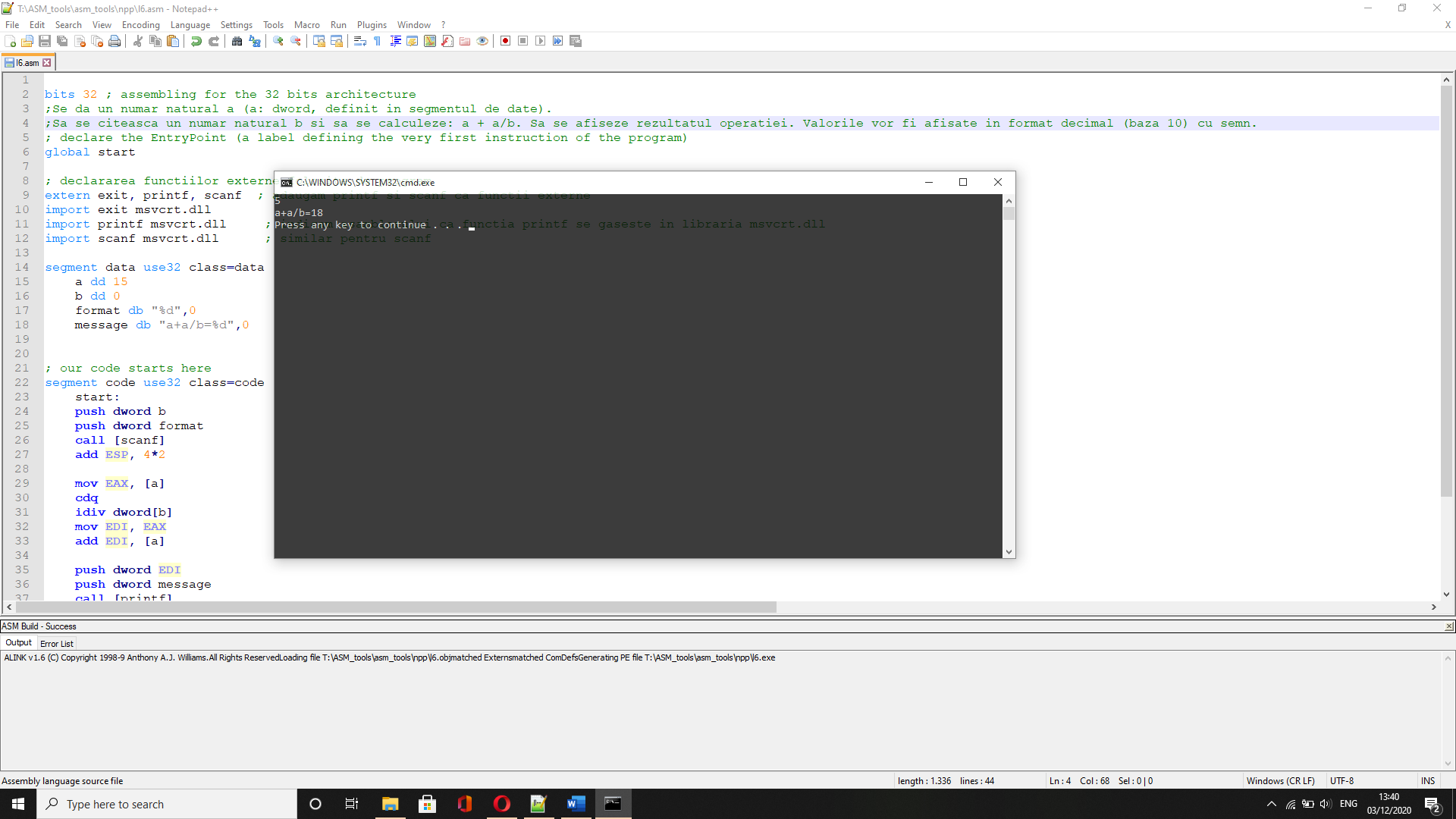
call [printf] ; print the message and the value of the expression

add ESP, 4\*2; clear the stack

; exit(0)

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

call [exit] ; call exit to terminate the program



22.

bits 32 ; assembling for the 32 bits architecture

;Se citesc de la tastatura doua numere a si b. Sa se calculeze valoarea expresiei (a+b)\*k, k fiind o constanta definita in segmentul de date. Afisati valoarea expresiei (in baza 10).

global start

; declararea functiilor externe folosite de program

extern exit, printf, scanf ; adaugam printf si scanf ca functii externe

import exit msvcrt.dll

import printf msvcrt.dll ; indicam asamblorului ca functia printf se gaseste in libraria msvcrt.dll

import scanf msvcrt.dll ; similar pentru scanf

segment data use32 class=data

a dd 0

b dd 0

k dd 10

format db "%d",0

message db "(a+b)\*k=%d",0

; our code starts here

segment code use32 class=code

start:

push dword a; push the value where I put the read number

push dword format; push its format

call [scanf] ; read the number from keyboard

add ESP, 4\*2; clear the stack

push dword b; push the value where I put the read number

push dword format; push its format

call [scanf] ; read the number from keyboard

add ESP, 4\*2; clear the stack

mov EDI, [a]; edi=a

add EDI, [b];edi=a+b

imul EDI, dword[k]; edi=(a+b)\*k

push dword EDI; push the value to be printed

push dword message; push its format

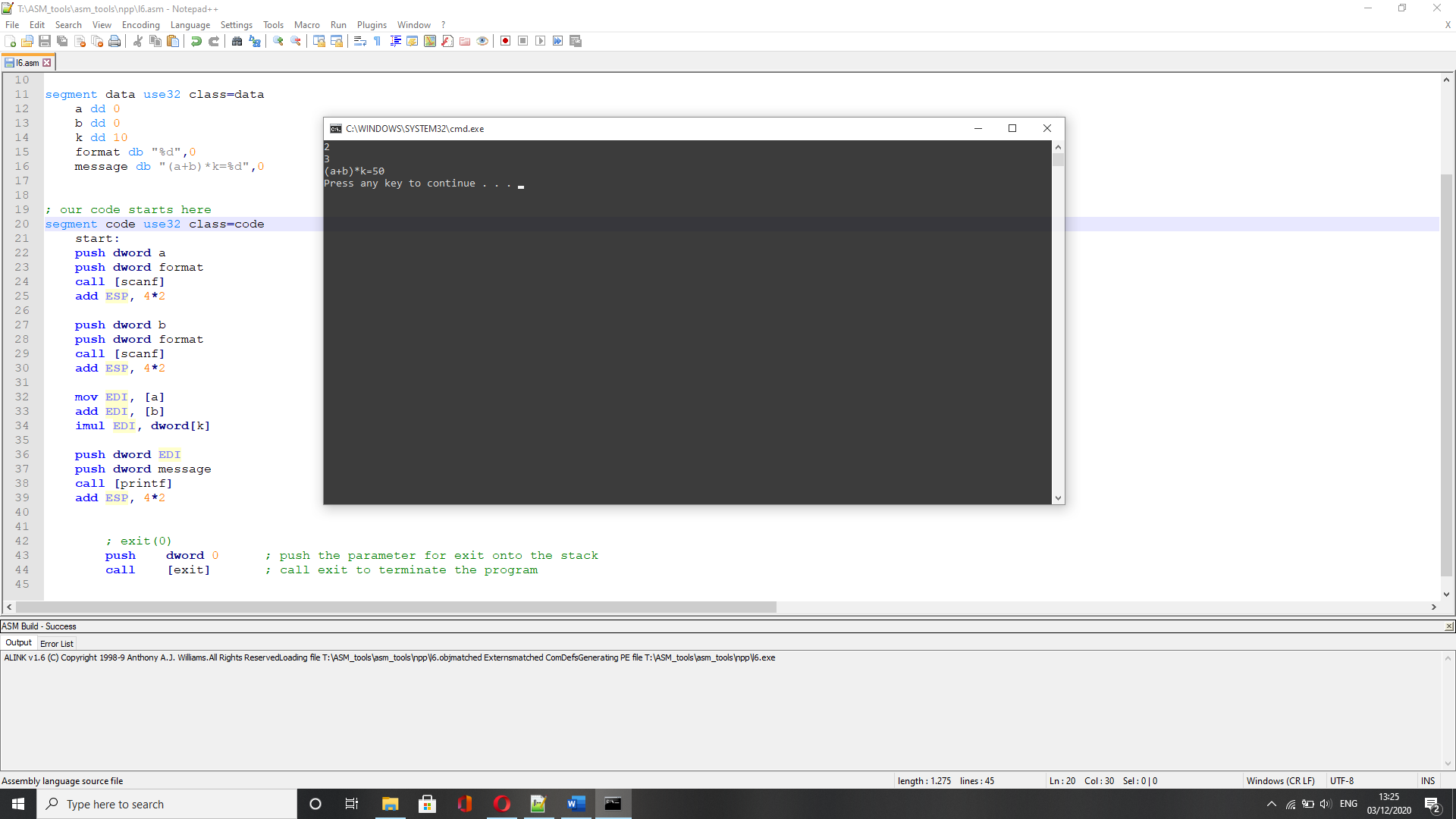
call [printf] ; print the message and the value of the expression

add ESP, 4\*2;clear the stack

; exit(0)

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

call [exit] ; call exit to terminate the program



8.

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, fprintf, fopen, fclose, printf, fread ; 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

import fprintf msvcrt.dll

import fopen msvcrt.dll

import fclose msvcrt.dll

import printf msvcrt.dll

import fread msvcrt.dll

;Se da un fisier text.

;Sa se citeasca continutul fisierului,

;sa se determine litera mare (uppercase) cu cea mai mare frecventa si sa se afiseze acea litera,

;impreuna cu frecventa acesteia. Numele fisierului text este definit in segmentul de date.

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

segment data use32 class=data

nume\_fisier db "lab10.txt", 0 ; numele fisierului

mod\_acces db "r", 0

descriptor dd -1 ; variabila in care se salveaza descriptorul fisierului

nr\_caract\_citite dd 0

len equ 100

buffer resb len

ascii resb 256

format db "Litera uppercase cu cea mai mare frecventa este %c. Frecventa literei %c este %d.", 0

; our code starts here

segment code use32 class=code

start:

push dword mod\_acces

push dword nume\_fisier

call [fopen]

add esp, 4 \* 2 ; se elibereaza parametrii de pe stiva

;se verifica daca functia fopen a deschis fisierul

cmp eax, 0

je final

;se salveaza descriptorul de fisier

mov [descriptor], eax

;se iau cate 100 de caractere din fisier si se transfera in buffer

repeta:

push dword [descriptor]

push dword len

push dword 1

push dword buffer

call[fread]

add esp, 4 \* 4

mov ecx, eax ; se contorizeaza numarul de caractrere pentru a parcurge buffer-ul intr-un loop

jecxz frecventa

;se parcurge buffer-ul

mov esi, buffer ; in esi se transfera sirul buffer

mov edi, ascii ; in edi se transfera sirul ascii

cld ; se parcurge sirul de la stanga la dreapta

parcurgere:

mov eax, 0

lodsb ; in al se transfera primul caracter din buffer

cmp al, 'A'

jb continua

cmp al, 'Z'

ja continua

inc dword [edi + eax]; se contorizeaza aparitia literei mari ce se afla in al

continua:

loop parcurgere

jmp repeta ; se citesc urmatoarele 100 de caractere

; se determina caracterul litera mare cu cea mai mare frecventa

frecventa:

push dword [descriptor]

call [fclose]

add esp, 4 \* 1

;se incepe determinare frecventei

mov bl, 0 ; se initializeaza numarul maxim de aparitii cu 0

mov ecx, 255 ; se initializeaza ecx cu numarul de caractere din tabela ascii

mov esi, ascii ; in esi se transfera sirul ascii(in care se cauta frecventele)

; se cauta litera mare cu numarul maxim de aparitii

aparitii:

lodsb ; in al se transfera frecventa caracterului din ascii

cmp al, bl ; verifica daca frecventa din al este mai mare decat vechiul numar maxim de aparitii

jb urmatorul ; daca frecventa din al este mai mica, se trece la urmatoarea frecventa

mov bl, al ; se salveaza in bl frecventa din al, daca este mai mare

mov eax, esi ; se salveaza in eax valoarea la care s-a ajuns in esi

sub eax, edi ; se cauta elementul in tabela de coduri ascii(+1)(edi pointeaza spre inceputul sirului ascii)

mov edx, eax ; se transfera in edx valoarea lui eax, adica caracterul cu numaraul maxim de aparitii(+1)

dec edx ; se scade 1 pentru a obtine caracterul cautat

urmatorul:

loop aparitii

; se afiseaza rezultatul calculat

mov al, bl; se transfera in al frecventa maxima salvata in bl

cbw ; se converteste valoarea la word

cwde; se converteste valoarea la dword

push dword eax ; frecventa

push dword edx ; caracterul

push dword edx

push dword format

call [printf]

add esp, 4 \* 3

final:

; exit(0)

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

call [exit] ; call exit to terminate the program

