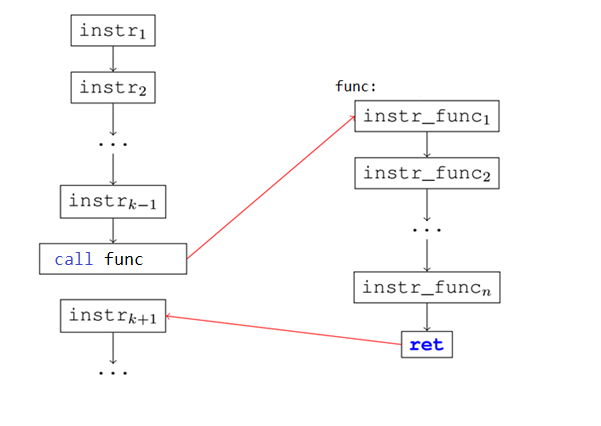
**Laborator 8 - Suport teoretic**

**Apeluri de funcţii sistem**

**Biblioteci de funcţii (function libraries)**

Chiar daca limbajul de asamblare foloseşte în mod direct componentele hardware ale sistemului, exista porțiuni de cod utilizate frecvent, care ar fi nepractic sa fie scrise de către programator de fiecare dată. De exemplu, comunicarea cu dispozitivele de intrare/ieşire, care presupune de cele mai multe ori protocoale complexe. Unul dintre rolurile sistemului de operare este acela de a abstractiza mașina hardware pentru programator, punând la dispoziție multiple biblioteci de funcţii ce pot fi apelate pentru anumite operații frecvente, cum ar fi afișarea datelor într-un anumit format, găsirea unui subşir într-un şir sau diverse funcţii matematice.

Utilizarea unei funcţii presupune saltul la porțiunea de cod corespunzătoare funcţiei, execuția acestui cod, urmata de revenirea la instrucţiunea de după cea care a apelat funcţia, ca în figura de mai jos:



## Utilizarea funcţiilor externe. Convenții de apel

Pentru apelul unei funcţii se folosește instrucţiunea call.

### *Sintaxa:*

call <nume\_funcţie\_sistem>

### *Semantica:*

Aceasta instrucţiune pune adresa instrucţiunii următoare pe stivă, apoi sare la începutul funcţiei apelate. Punerea pe stivă a adresei instrucţiunii următoare (adresa de revenire) se face pentru ca la finalul execuției să se poată reveni la codul apelant.

## Apelarea unei funcţii sistem. Parametrii

De cele mai multe ori, funcţiile pot primi parametri, şi pot întoarce rezultate. Există mai multe convenții pentru a face aceste lucruri, dintre care in acest laborator vom folosi convenția **cdecl**.

O convenție de apel nu ţine de sintaxa limbajului de asamblare, ci este un ”contract” între autorul funcţiei şi utilizatorii acesteia, ce specifică modul de transmitere a parametrilor, respectiv de întoarcere a rezultatului.

### Convenția cdecl

* Argumentele se pun pe stivă de la dreapta la stânga
* Rezultatul implicit este returnat de către funcţia sistem în EAX
* Regiştrii EAX, ECX, EDX pot fi folosiți în interiorul funcţiei (îşi pot modifica valoarea!)
* Atenție deci, dacă doriți păstrarea valorilor inițiale din EAX, ECX, EDX sa salvați aceste valori (în variabile de memorie, pe stivă sau în alți regiştri) înainte de apelul funcţiei
* Funcţia nu va elibera argumentele de pe stivă. Este responsabilitatea programatorului să facă acest lucru după apelul funcţiei
* Exemple: printf, scanf

## Funcţii standard din msvcrt

### Afișarea pe ecran

Pentru afișarea unui text pe ecran, ce respecta un anumit format, se folosește funcţia printf.

Sintaxa funcţiei printf în limbaj de programare de nivel înalt este:

int printf(const char \* format, variabila\_1, constanta\_2, ...);

Funcţia printf respecta convenția cdecl.

Primul argument al funcţiei este un şir de caractere ce conține formatul afișării, urmat de un număr de argumente (valori constante sau nume de variabile) egal cu cel specificat in cadrul formatului.

Şirul de caractere transmis în parametrul format poate conține anumite marcaje de formatare, ce încep cu caracterul ’%’, care vor fi înlocuite de valorile specificate în următoarele argumente, formatate corespunzător.

|  |  |  |  |
| --- | --- | --- | --- |
| **Specificator** | **Ce se afișează** | **Exemplu** | **Dimensiune reprezentare valoare** |
| c | Caracter | a | byte |
| d sau i | Întreg zecimal cu semn | 392 | dword |
| u | Întreg zecimal fără semn | 7235 | dword |
| x | Număr in hexazecimal fără semn | 7fa | dword |
| s | String (şir de caractere, terminat cu 0) | exemplu | şir de bytes terminat in 0 |

### La intrarea in corpul funcției printf, in vârful stivei se afla un dublucuvânt cu Adresa de Revenire, apoi imediat sub el se afla alt dublucuvânt care va conține offsetul stringului de formatare – argument al funcției printf. Sub acest dublucuvânt se vor afla alte dublucuvinte corespunzătoare celorlalte argumente ale funcției printf. Descriptorii de afișare %d, %i, %c, %x, %u în cazul funcției printf vor fi aplicați totdeauna unui dublucuvânt din stivă (chiar şi în cazul lui %c care folosește şi el un dublucuvânt de pe stivă, dar din acest dublucuvânt consideră doar octetul cel mai puțin semnificativ). Astfel, primul descriptor de afișare se va aplica celui de-al treilea dublucuvânt de pe stivă (numărat fata de vârful stivei). Cel de-al doilea eventual descriptor se va aplica celui de-al patrulea dublucuvânt de pe stivă (numărat faţă de vârful stivei), cel de-al treilea eventual descriptor din stringul de formatare se va aplica celui de-al cincilea dublucuvânt de pe stivă (numărat faţă de vârful stivei) şi tot așa.

#### Exemple:

##### Afișarea unui mesaj

* În limbaj de programare de nivel înalt:

printf("Ana are mere");

* Echivalentul în limbaj de asamblare:

segment data use32 class=data

mesaj db "Ana are mere", 0 ; definim mesajul

segment code use32 class=code

; ...

push dword mesaj ; punem parametrul pe stiva

call [printf] ; apelam funcţia printf

add esp, 4 \* 1 ; eliberam parametrii de pe stiva

; ...

##### Afișarea unui număr natural cu semn in baza 10

* În limbaj de programare de nivel înalt:

printf("%d", -17);

* Echivalentul în limbaj de asamblare:

segment data use32 class=data

format db "%d", 0 ; definim formatul

segment code use32 class=code

; ...

push dword -17 ; punem parametrii pe stiva de la dreapta la stânga

push dword format

call [printf] ; apelam funcţia printf

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

; ...

##### Afișarea unui număr natural in baza 16

* În limbaj de programare de nivel înalt:

printf("%x", 0xAB);

* Echivalentul în limbaj de asamblare:

segment data use32 class=data

format db "%x", 0 ; definim formatul

segment code use32 class=code

; ...

push dword 0xAB ; punem parametrii pe stivă de la dreapta la stânga

push dword format

call [printf] ; apelăm funcţia printf

add esp, 4 \* 2 ; eliberăm parametrii de pe stivă

; ...

##### Afișarea unui mesaj care conține un număr natural în baza 10, stocat intr-o variabila n

* În limbaj de programare de nivel înalt:

printf("Ana are %d mere", n);

* Echivalentul în limbaj de asamblare:

segment data use32 class=data

n dd 7

format db "Ana are %d mere", 0 ; definim formatul

segment code use32 class=code

; ...

push dword [n] ; punem pe stiva valoarea lui n

push dword format

call [printf] ; apelam funcţia printf

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

; ...

##### Afișarea unui mesaj care conține mai multe numere in baza 10.

* În limbaj de programare de nivel înalt:

printf("Ana are %d mere şi %d pere", 7, 8);

* Echivalentul în limbaj de asamblare:

segment data use32 class=data

format db "Ana are %d mere şi %d pere", 0 ; definim formatul

segment code use32 class=code

; ...

push dword 8 ; punem parametrii pe stiva

push dword 7

push dword format

call [printf] ; apelam funcţia printf

add esp, 4 \* 3 ; eliberam parametrii de pe stivă

; ...

### Citirea de la tastatura

Pentru a citi date de la tastatura se folosește funcţia scanf.

Sintaxa funcţiei scanf în limbaj de programare de nivel înalt este:

int scanf(const char \* format, adresa\_variabila\_1, ...);

Sintaxa acestei funcţii este similara cu cea a funcţiei printf. Diferența majoră constă în faptul că argumentele sale nu trebuie să fie valori constante sau conținuturi de variabile, ci numai adrese în memorie, unde se vor stoca valorile citite.

#### Exemplu:

##### Citirea unui număr natural în variabila n

* În limbaj de programare de nivel înalt:

scanf("%d", &n);

&n reprezintă adresa variabilei n in care funcţia scanf va completa valoarea citită de la tastatura

* Echivalentul în limbaj de asamblare:

segment data use32 class=data

n dd 0 ; definim variabila n

format db "%d", 0 ; definim formatul

segment code use32 class=code

; ...

push dword n ; punem parametrii pe stivă de la dreapta la stânga

push dword format

call [scanf] ; apelăm funcţia scanf pentru citire

add esp, 4 \* 2 ; eliberăm parametrii de pe stivă

; ...

**Operații cu fișiere text**

**Operații cu fişiere text**

Un fişier reprezintă o secvență de octeţi. Pentru a citi dintr-un fişier sau pentru a scrie într-un fişier, e nevoie de 3 paşi:

1. Deschiderea fişierului, care poate consta în:
   * Deschiderea unui fişier existent
   * Crearea unui fişier nou
2. Efectuarea operațiilor de scriere şi/sau citire
3. Închiderea fişierului.

## Deschiderea unui fişier

Pentru deschiderea unui fişier existent sau crearea unui fişier nou, se folosește funcţia **fopen**

### *Sintaxa funcţiei*

**fopen** în limbaj de programare de nivel înalt este:

FILE \* fopen(const char\* nume\_fişier, const char \* mod\_acces)

Funcţia fopen respecta convenția cdecl şi se găsește în msvcrt.dll .

### *Argumentele funcţiei fopen:*

Primul argument al funcţiei este adresa unui şir de caractere reprezentând numele fişierului: Al doilea argument este adresa unui şir de caractere, reprezentând modalitatea in care se va deschide fişierul.

|  |  |  |
| --- | --- | --- |
| Mod | Semnificație | Descriere |
| r | citire (read) | -    Deschide un fişier text pentru citire. Fişierul trebuie sa existe deja pe disc. |
| w | scriere (write) | -    Dacă nu exista un fişier cu acel nume, creează fişierul şi îl deschide pentru scriere. -    Dacă un fişier cu acel nume exista deja, deschide acel fişier pentru scriere. Conținutul inițial va fi șters. Scrierea se va face de la începutul fişierului. |
| a | adaugare (append) | -    Daca nu exista un fişier cu acel nume, creează fişierul şi îl deschide pentru scriere. -    Daca un fişier cu acel nume exista deja, deschide acel fişier pentru scriere, dar scrierea se va face la sfârșitul fişierului, in continuarea conținutului existent. Conținutul inițial al fişierului va fi păstrat. |
| r+ | citire şi scriere fişier existent | -    Deschide un fişier text pentru citire şi scriere. Fişierul trebuie să existe deja pe disc. |
| w+ | citire şi scriere | -    Dacă nu exista un fişier cu acel nume, creează fişierul şi îl deschide pentru citire şi scriere.  -    Dacă un fişier cu acel nume există deja, deschide acel fişier pentru citire şi scriere. Conținutul inițial va fi șters. Scrierea se va face de la începutul fişierului. |
| a+ | citire şi adăugare | -    Dacă nu există un fişier cu acel nume, creează fişierul şi îl deschide pentru citire şi scriere.  -    Dacă un fişier cu acel nume există deja, deschide acel fişier pentru citire şi scriere. Conținutul inițial al fişierului va fi păstrat. Citirea se va face de la începutul fişierului. Scrierea se va face in continuarea conținutului existent. |

### *Observații:*

* Numele unui fişier trebuie sa includă şi extensia (ex: nume.txt, exemplu.asm).
* Fişierele se vor crea sau deschide din directorul curent (în același director in care se afla fişierul sursa asm). Important: pentru a putea deschide un fişier existent folosind numele acestuia, fişierul trebuie sa se afle in același director cu fişierul sursa .asm, altfel acesta nu va fi găsit.
* Operațiile de scriere nu vor reuși pentru fişiere deschise doar cu drepturi de citire (ex: ”r”). Operațiile de citire nu vor reuși pentru fişiere deschise doar cu drepturi de scriere sau adăugare (ex: ”w”, ”a”)
* Ambele argumente ale funcţiei fopen reprezintă şiruri de caractere care trebuie sa se termine cu valoarea 0 (asemenea formatului pentru funcţia printf).

#### Valoarea returnata de funcţia fopen:

Daca fişierul este deschis cu succes, funcţia fopen va completa in registrul EAX un identificator (descriptor de fişier) care va fi folosit in continuare pentru a lucra cu acel fişier (pentru operații de scriere, citire, etc.). Altfel (in caz de eroare), funcţia fopen va completa in registrul EAX valoarea 0.

### *Alte observații:*

Este important sa se verifice valoarea returnata de funcţie in EAX (daca nu a fost eroare), înainte de a efectua alte operații cu acel fişier. Daca in cadrul unui program se deschid mai multe fişiere diferite folosind funcţia fopen, fiecare valoare returnata de funcţiei trebuie salvata separat, deoarece reprezintă o valoare distincta prin care este identificat un fişier. După finalizarea lucrului cu un fişier deschis, este important să se şi închidă acel fişier (de obicei se face la finalul programului – înainte de exit). Pentru a închide un fişier (deschis in prealabil de funcţia fopen) se folosește funcţia fclose.

## Scrierea intr-un fişier

Pentru a scrie un text intr-un fişier se folosește funcţia **fprintf**.

### *Sintaxa funcţiei*

**fprintf** în limbaj de programare de nivel înalt este:

int fprintf(FILE \* stream, const char \* format, <variabila\_1>, <constanta\_2>, <...>)

Funcţia fprintf respecta convenția cdecl şi se găsește in msvcrt.dll . Sintaxa funcţiei fprintf este asemănătoare cu sintaxa funcţiei printf (folosită pentru afișare în consolă). Diferența este faptul ca funcţia fprintf are ca prim parametru identificatorul fişierului în care se va scrie textul, în plus faţă de parametrii funcţiei printf.

### *Argumentele funcţiei fprintf*

Primul argument al funcţiei reprezintă descriptorul de fişier (identificatorul) returnat de apelul funcţiei fopen. Următorul argument al funcţiei este un şir de caractere ce conține formatul afișării, urmat de un număr de argumente (valori constante sau nume de variabile) egal cu cel specificat în cadrul formatului. Asemenea funcţiei printf, şirul de caractere transmis in parametrul format poate conține anumite marcaje de formatare, ce încep cu caracterul ’%’, care vor fi înlocuite de valorile specificate în următoarele argumente, formatate corespunzător.

|  |  |  |  |
| --- | --- | --- | --- |
| Specificator | Ce se afișează | Exemplu | Dimensiune reprezentare valoare |
| c | Caracter | a | byte |
| d sau i | Întreg zecimal cu semn | 392 | dword |
| u | Întreg zecimal fără semn | 7235 | dword |
| x | Număr in hexazecimal fără semn | 7fa | dword |
| s | String (şir de caractere, terminat cu 0) | exemplu | şir de bytes terminat in 0 |

## Citirea dintr-un fişier

Pentru a citi un text dintr-un fişier se folosește funcţia **fread**.

### *Sintaxa funcţiei*

**fread** în limbaj de programare de nivel înalt este:

int fread(void \* str, int size, int count, FILE \* stream)

Funcţia fread respecta conventia cdecl şi se gaseste in msvcrt.dll .

### *Argumentele funcţiei fread*

Primul argument al funcţiei fread reprezintă adresa unui şir de elemente în care se vor completa datele citite din fişier. Al doilea argument reprezintă dimensiunea unui element care va fi citit din fişier. Al treilea argument reprezintă numărul maxim de elemente care se vor citi din fişier. Ultimul argument al funcţiei reprezintă descriptorul de fişier (identificatorul) returnat de apelul funcţiei fopen. În cazul citirii fişierelor text, primul argument al funcţiei fread este un şir de bytes şi al doilea argument este 1 (=dimensiunea unui byte). Al treilea argument este dimensiunea şirului de bytes (numărul de elemente).

### *Valoarea returnata de funcţia fread:*

Funcţia fread va completa în registrul EAX numărul de elemente citite. Dacă acest număr este mai mic decât valoarea argumentului count, atunci fie apelul funcţiei fread a întâmpinat o eroare la citire, fie s-a ajuns la finalul fişierului.

### *Observații:*

Fişierele text pot fi avea dimensiuni prea mari pentru putea citi conținutul acestora cu un singur apel al funcţiei fread. În acest caz este nevoie de apeluri repetate ale funcţiei fread, până când întreg conținutul fişierului este citit. În secţiunea „Exemple” vom prezenta un program care exemplifica acest scenariu. Pentru a verifica dacă s-a ajuns la finalul fişierului cu operația de citire se poate verifica dacă valoarea returnată de fread este 0.

## Închiderea unui fişier deschis

După finalizarea lucrului cu un fişier deschis, acesta trebuie închis. Acest pas nu trebuie să lipsească dintr-un program care a deschis fişiere. Pentru închiderea unui fişier se folosește funcţia **fclose**.

### *Sintaxa funcţiei*

**fclose** în limbaj de programare de nivel înalt este:

int fclose(FILE \* descriptor)

Funcţia fclose respecta convenția cdecl şi se găsește în msvcrt.dll .

### *Argumentul funcţiei fclose*

Argumentul funcţiei fclose este descriptorul de fişier (identificatorul) returnat de apelul funcţiei fopen.

**EXEMPLE**

1. Atenție la dimensiunea de reprezentare a variabilelor în funcție de formatul dorit. Vezi tabel la secţiunea [Funcţii standard din msvcrt](http://www.cs.ubbcluj.ro/~vancea/asc/lab8-teorie.php#ui-id-7) din Teorie. Urmăriţi exemplul de mai jos care e GREŞIT (greșeală frecventă) şi spuneți ce se întâmplă:

Ne propunem ca programul de mai jos să citească de la tastatură un număr şi să afișeze pe ecran valoarea numărului citit împreună cu un mesaj.

bits 32

global start

; declararea funcţiilor externe folosite de program

extern exit, printf, scanf ; adăugăm printf şi scanf ca funcţii externe

import exit msvcrt.dll

import printf msvcrt.dll ; indicăm asamblorului că funcţia printf se găseşte în biblioteca msvcrt.dll

import scanf msvcrt.dll ; similar pentru scanf

segment data use32 class=data

n db 0 ; în această variabilă se va stoca valoarea citită de la tastatură

message db "Numărul citit este n= %d", 0

format db "%d", 0 ; %d <=> un număr zecimal (baza 10)

segment code use32 class=code

start:

; vom apela scanf(format, n) => se va citi un număr în variabila n

; punem parametrii pe stivă de la dreapta la stânga

push dword n ; ! adresa lui n, nu valoarea

push dword format

call [scanf] ; apelăm funcţia scanf pentru citire

add esp, 4 \* 2 ; eliberam parametrii de pe stiva; 4 = dimensiunea unui dword; 2 = nr de parametri

;convertim n la dword pentru a pune valoarea pe stiva

mov eax,0

mov al,[n]

;afişăm mesajul şi valoarea lui n

push eax

push dword message

call [printf]

add esp,4\*2

; exit(0)

push dword 0 ; punem pe stiva parametrul pentru exit

call [exit] ; apelam exit pentru a încheia programul

În exemplul de mai sus se suprascrie începutul din mesaj şi nu se mai afișează nimic pe ecran.

2. ; Codul de mai jos va afisa mesajul "Ana are 17 mere"

bits 32

global start

; declararea funcţiilor externe folosite de program

extern exit, printf ; adaugam printf ca funcţie externa

import exit msvcrt.dll

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

segment data use32 class=data

; sirurile de caractere sunt de tip byte

format db "Ana are %d mere", 0 ; %d va fi inlocuit cu un număr

; sirurile de caractere pt funcţiile C trebuie terminate cu valoarea 0

segment code use32 class=code

start:

mov eax, 17

; vom apela printf(format, 17) => se va afisa: „Ana are 17 mere”

; punem parametrii pe stiva de la dreapta la stanga

push dword eax

push dword format ; ! pe stiva se pune adresa string-ului, nu valoarea

call [printf] ; apelam funcţia printf pentru afisare

add esp, 4 \* 2 ; eliberam parametrii de pe stiva; 4 = dimensiunea unui dword; 2 = nr de parametri

; exit(0)

push dword 0 ; punem pe stiva parametrul pentru exit

call [exit] ; apelam exit pentru a incheia programul

3. Codul de mai jos va afisa mesajul ”n=”, apoi va citi de la tastatura valoarea numărului n.

bits 32

global start

; declararea funcţiilor externe folosite de program

extern exit, printf, scanf ; adaugam printf şi scanf ca funcţii externa

import exit msvcrt.dll

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

import scanf msvcrt.dll ; similar pentru scanf

segment data use32 class=data

n dd 0 ; in aceasta variabila se va stoca valoarea citita de la tastatura

; sirurile de caractere sunt de tip byte

message db "n=", 0 ; sirurile de caractere pentru funcţiile C trebuie sa se termine cu valoarea 0 (nu caracterul)

format db "%d", 0 ; %d <=> un număr decimal (baza 10)

segment code use32 class=code

start:

; vom apela printf(message) => se va afisa "n="

; punem parametrii pe stiva

push dword message ; ! pe stiva se pune adresa string-ului, nu valoarea

call [printf] ; apelam funcţia printf pentru afisare

add esp, 4\*1 ; eliberam parametrii de pe stiva ; 4 = dimensiunea unui dword; 1 = nr de parametri

; vom apela scanf(format, n) => se va citi un număr in variabila n

; punem parametrii pe stiva de la dreapta la stanga

push dword n ; ! adresa lui n, nu valoarea

push dword format

call [scanf] ; apelam funcţia scanf pentru citire

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

; 4 = dimensiunea unui dword; 2 = nr de parametri

; exit(0)

push dword 0 ; punem pe stiva parametrul pentru exit

call [exit] ; apelam exit pentru a incheia programul

4. Codul de mai jos va calcula rezultatul unor operatii aritmetice in registrul EAX, va salva valoarea registrilor, apoi va afisa valoarea rezultatului şi va restaura valoarea registrilor.

bits 32

global start

; declararea funcţiilor externe folosite de program

extern exit, printf ; adaugam printf ca funcţie externa

import exit msvcrt.dll

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

segment data use32 class=data

; sirurile de caractere sunt de tip byte

format db "%d", 0 ; %d <=> un număr decimal (baza 10)

segment code use32 class=code

start:

; vom calcula 20 + 123 + 7 in EAX

mov eax, 20

add eax, 123

add eax, 7 ; eax = 150 (baza 10) sau 0x96 (baza 16)

; salvam valoarea registrilor deoarece apelul funcţiei sistem printf va modifica valoarea acestora

; folosim instructiunea PUSHAD care salveaza pe stiva valorile mai multor registrii, printre care EAX, ECX, EDX şi EBX

; in acest exemplu este important sa salvam doar valoarea registrului EAX, dar instructiunea poate fi aplicata generic

PUSHAD

; vom apela printf(format, eax) => vom afisa valoarea din eax

; punem parametrii pe stiva de la dreapta la stanga

push dword eax

push dword format ; ! pe stiva se pune adresa string-ului, nu valoarea

call [printf] ; apelam funcţia printf pentru afisare

add esp, 4\*2 ; eliberam parametrii de pe stiva ; 4 = dimensiunea unui dword; 2 = nr de parametri

; dupa apelul funcţiei printf registrul EAX are o valoare setata de aceasta funcţie (nu valoarea 150 pe care am calculat-o la inceputul programului)

; restauram valoarea registrilor salvati pe stiva la apelul instructii PUSHAD folosind instructiunea POPAD

; aceasta instructiune ia valori de pe stiva şi le completeaza in mai multi registrii printre care EAX, ECX, EDX şi EBX

; este important ca inaintea unui apel al instructiunii POPAD sa ne asiguram ca exista suficiente valori

; pe stiva pentru a fi incarcate in registrii (de exemplu ca anterior a fost apelata instructiunea PUSHA)

POPAD

; acum valoarea registrului EAX a fost restaurata la valoarea de dinaintea apelului instructiunii PUSHAD (in acest caz valoarea 150)

; exit(0)

push dword 0 ; punem pe stiva parametrul pentru exit

call [exit] ; apelam exit pentru a incheia programul

**Exemple mari**

1. Codul de mai jos va crea un fişier gol, numit "ana.txt" in directorul curent.

Programul va folosi funcţia fopen pentru deschiderea/crearea fişierului şi funcţia fclose pentru închiderea fişierului creat.

Deoarece în apelul funcţiei fopen programul folosește modul de acces "w", daca un fişier cu același nume exista deja in directorul curent, conținutul acelui fişier va fi șters. Detalii despre modurile de acces sunt prezentate in secţiunea "Suport teoretic"

bits 32

global start

; declare external functions needed by our program

extern exit, fopen, fclose

import exit msvcrt.dll

import fopen msvcrt.dll

import fclose msvcrt.dll

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

segment data use32 class=data

nume\_fişier db "ana.txt", 0 ; numele fişierului care va fi creat

mod\_acces db "w", 0 ; modul de deschidere a fişierului -

; w - pentru scriere. daca fişierul nu exista, se va crea

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

; our code starts here

segment code use32 class=code

start:

; apelam fopen pentru a crea fişierul

; funcţia va returna in EAX descriptorul fişierului sau 0 in caz de eroare

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

push dword mod\_acces

push dword nume\_fişier

call [fopen]

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

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

; verificam daca funcţia fopen a creat cu succes fişierul (daca EAX != 0)

cmp eax, 0

je final

; apelam funcţia fclose pentru a inchide fişierul

; fclose(descriptor\_fis)

push dword [descriptor\_fis]

call [fclose]

add esp, 4

final:

; exit(0)

push dword 0

call [exit]

2. Codul de mai jos va crea un fişier numit "ana.txt" in directorul curent şi va scrie un text in acel fişier.

; Programul va folosi functia fopen pentru deschiderea/crearea fişierului, functia fprintf pentru scrierea in fişier şi functia fclose pentru inchiderea fişierului creat.

; Deoarece in apelul functiei fopen programul foloseste modul de acces "w", daca un fişier cu acelasi nume exista deja in directorul curent, continutul acelui fişier va fi suprascris. Detalii despre modurile de acces sunt prezentate in secţiunea "Suport teoretic"

bits 32

global start

; declare external functions needed by our program

extern exit, fopen, fprintf, fclose

import exit msvcrt.dll

import fopen msvcrt.dll

import fprintf msvcrt.dll

import fclose msvcrt.dll

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

segment data use32 class=data

nume\_fişier db "ana.txt", 0 ; numele fişierului care va fi creat

mod\_acces db "w", 0 ; modul de deschidere a fişierului -

; w - pentru scriere. daca fiserul nu exista, se va crea

text db "Ana are mere.", 0 ; textul care va fi scris in fişier

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

; our code starts here

segment code use32 class=code

start:

; apelam fopen pentru a crea fişierul

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

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

push dword mod\_acces

push dword nume\_fişier

call [fopen]

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

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

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

cmp eax, 0

je final

; scriem textul in fişierul deschis folosind functia fprintf

; fprintf(descriptor\_fis, text)

push dword text

push dword [descriptor\_fis]

call [fprintf]

add esp, 4\*2

; apelam functia fclose pentru a inchide fişierul

; fclose(descriptor\_fis)

push dword [descriptor\_fis]

call [fclose]

add esp, 4

final:

; exit(0)

push dword 0

call [exit]

3. ; Codul de mai jos va deschide un fişier numit "ana.txt" in directorul curent şi va adauga un text la finalul acelui fişier.

; Programul va folosi functia fopen pentru deschiderea/crearea fişierului, functia fprintf pentru scrierea in fişier şi functia fclose pentru inchiderea fişierului creat.

; Deoarece in apelul functiei fopen programul foloseste modul de acces "a", daca un fişier cu numele exista deja in directorul curent, la continutul acelui fişier se va adauga un text. Daca fişierul cu numele dat nu exista, se va crea un fişier nou cu acel nume şi se va scrie textul in fişier. Detalii despre modurile de acces sunt prezentate in secţiunea "Suport teoretic"

bits 32

global start

; declare external functions needed by our program

extern exit, fopen, fprintf, fclose

import exit msvcrt.dll

import fopen msvcrt.dll

import fprintf msvcrt.dll

import fclose msvcrt.dll

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

segment data use32 class=data

nume\_fişier db "ana.txt", 0 ; numele fişierului care va fi creat

mod\_acces db "a", 0 ; modul de deschidere a fişierului -

; a - pentru adaugare. daca fiserul nu exista, se va crea

text db "Ana are şi pere.", 0 ; textul care va fi adaugat in fişier

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

; our code starts here

segment code use32 class=code

start:

; apelam fopen pentru a crea fişierul

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

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

push dword mod\_acces

push dword nume\_fişier

call [fopen]

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

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

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

cmp eax, 0

je final

; adaugam/scriem textul in fişierul deschis folosind functia fprintf

; fprintf(descriptor\_fis, text)

push dword text

push dword [descriptor\_fis]

call [fprintf]

add esp, 4\*2

; apelam functia fclose pentru a inchide fişierul

; fclose(descriptor\_fis)

push dword [descriptor\_fis]

call [fclose]

add esp, 4

final:

; exit(0)

push dword 0

call [exit]

4. Codul de mai jos va deschide un fişier numit "ana.txt" din directorul curent şi va citi un text de maxim 100 de caractere din acel fişier.

Programul va folosi funcția fopen pentru deschiderea fişierului, funcția fread pentru citirea din fişier şi funcția fclose pentru închiderea fişierului deschis.

Deoarece in apelul functiei fopen programul folosește modul de acces "r", daca un fişier cu numele dat nu exista in directorul curent, apelul funcției fopen nu va reuși (eroare). Detalii despre modurile de acces sunt prezentate in secţiunea "Suport teoretic".

bits 32

global start

; declare external functions needed by our program

extern exit, fopen, fread, fclose

import exit msvcrt.dll

import fopen msvcrt.dll

import fread msvcrt.dll

import fclose msvcrt.dll

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

segment data use32 class=data

nume\_fişier db "ana.txt", 0 ; numele fişierului care va fi deschis

mod\_acces db "r", 0 ; modul de deschidere a fişierului -

; r - pentru scriere. fişierul trebuie sa existe

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

len equ 100 ; numărul maxim de elemente citite din fişier.

text times len db 0 ; sirul in care se va citi textul din fişier

; our code starts here

segment code use32 class=code

start:

; apelam fopen pentru a deschide fişierul

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

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

push dword mod\_acces

push dword nume\_fişier

call [fopen]

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

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

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

cmp eax, 0

je final

; citim textul in fişierul deschis folosind functia fread

; eax = fread(text, 1, len, descriptor\_fis)

push dword [descriptor\_fis]

push dword len

push dword 1

push dword text

call [fread]

add esp, 4\*4 ; dupa apelul functiei fread EAX contine numărul de caractere citite din fişier

; apelam functia fclose pentru a inchide fişierul

; fclose(descriptor\_fis)

push dword [descriptor\_fis]

call [fclose]

add esp, 4

final:

; exit(0)

push dword 0

call [exit]

5. Codul de mai jos va deschide un fişier numit "ana.txt" din directorul curent, va citi un text scurt din acel fişier, apoi va afisa in consola numărul de caractere citite şi textul citit din fişier.

Programul va folosi functia fopen pentru deschiderea fişierului, functia fread pentru citirea din fişier şi functia fclose pentru inchiderea fişierului creat.

Deoarece in apelul functiei fopen programul foloseste modul de acces "r", daca un fişier numele dat nu exista in directorul curent, apelul functiei fopen nu va reusi (eroare). Detalii despre modurile de acces sunt prezentate in secţiunea "Suport teoretic".

; In acest program sirul de caractere in care se va citi textul din fişier trebuie sa aiba o lungime cu 1 mai mare decat numărul maxim de elemente care vor fi citite din fişier deoarece acest sir va fi afisat in consola folosind functia printf.

; Orice sir de caractere folosit de functia printf trebuie sa fie terminat in 0, altfel afisarea nu va fi corecta.

; Daca fişierul ar contine mai mult de <len> caractere şi dimensiunea sirului destinatie era exact <len>, intregul sir ar fi fost completat cu valori citite din fişier, astfel sirul nu se mai termina cu valoarea 0.

bits 32

global start

; declare external functions needed by our program

extern exit, fopen, fread, fclose, printf

import exit msvcrt.dll

import fopen msvcrt.dll

import fread msvcrt.dll

import fclose msvcrt.dll

import printf msvcrt.dll

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

segment data use32 class=data

nume\_fişier db "ana.txt", 0 ; numele fişierului care va fi creat

mod\_acces db "r", 0 ; modul de deschidere a fişierului -

; r - pentru scriere. fişierul trebuie sa existe

len equ 100 ; numărul maxim de elemente citite din fişier.

text times (len+1) db 0 ; sirul in care se va citi textul din fişier (dimensiune len+1 explicata mai sus)

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

format db "Am citit %d caractere din fişier. Textul este: %s", 0 ; formatul - utilizat pentru afisarea textului citit din fişier

; %s reprezinta un sir de caractere

; our code starts here

segment code use32 class=code

start:

; apelam fopen pentru a deschide fişierul

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

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

push dword mod\_acces

push dword nume\_fişier

call [fopen]

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

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

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

cmp eax, 0

je final

; citim textul in fişierul deschis folosind functia fread

; eax = fread(text, 1, len, descriptor\_fis)

push dword [descriptor\_fis]

push dword len

push dword 1

push dword text

call [fread]

add esp, 4\*4 ; dupa apelul functiei fread EAX contine numărul de caractere citite din fişier

; afisam numărul de caractere citite şi textul citit

; printf(format, eax, text)

push dword text

push dword EAX

push dword format

call [printf]

add esp, 4\*3

; apelam functia fclose pentru a inchide fişierul

; fclose(descriptor\_fis)

push dword [descriptor\_fis]

call [fclose]

add esp, 4

final:

; exit(0)

push dword 0

call [exit]

6. Codul de mai jos va deschide un fişier numit "input.txt" din directorul curent şi va citi intregul text din acel fişier, in etape, cate 100 de caractere intr-o etapa.

Deoarece un fişier text poate fi foarte lung, nu este intotdeauna posibil sa citim fişierul intr-o singura etapa pentru ca nu putem defini un sir de caractere suficient de lung pentru intregul text din fişier. De aceea, prelucrarea fişierelor text in etape este necesara.

Programul va folosi functia fopen pentru deschiderea fişierului, functia fread pentru citirea din fişier şi functia fclose pentru inchiderea fişierului creat.

Deoarece in apelul functiei fopen programul foloseste modul de acces "r", daca un fişier numele dat nu exista in directorul curent, apelul functiei fopen nu va reusi (eroare). Detalii despre modurile de acces sunt prezentate in secţiunea "Suport teoretic".

bits 32

global start

; declare external functions needed by our program

extern exit, fopen, fclose, fread

import exit msvcrt.dll

import fopen msvcrt.dll

import fread msvcrt.dll

import fclose msvcrt.dll

; our data is declared here

segment data use32 class=data

nume\_fişier db "input.txt", 0 ; numele fişierului care va fi deschis

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

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

nr\_car\_citite dd 0 ; variabila in care vom salva numărul de caractere citit din fişier in etapa curenta

len equ 100 ; numărul maxim de elemente citite din fişier intr-o etapa

buffer resb len ; sirul in care se va citi textul din fişier

; our code starts here

segment code use32 class=code

start:

; apelam fopen pentru a deschide fişierul

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

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

push dword mod\_acces

push dword nume\_fişier

call [fopen]

add esp, 4\*2

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

cmp eax, 0

je final

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

; echivalentul in pseudocod al urmatoarei secvente de cod este:

;repeta

; nr\_car\_citite = fread(buffer, 1, len, descriptor\_fis)

; daca nr\_car\_citite > 0

; ; instructiuni pentru procesarea caracterelor citite in aceasta etapa

;pana cand nr\_car\_citite == 0

bucla:

; citim o parte (100 caractere) din textul in fişierul deschis folosind functia fread

; eax = fread(buffer, 1, len, descriptor\_fis)

push dword [descriptor\_fis]

push dword len

push dword 1

push dword buffer

call [fread]

add esp, 4\*4

; eax = număr de caractere / bytes citite

cmp eax, 0 ; daca numărul de caractere citite este 0, am terminat de parcurs fişierul

je cleanup

mov [nr\_car\_citite], eax ; salvam numărul de caractere citie

; instructiunile pentru procesarea caracterelor citite in aceasta etapa incep aici

; ...

; reluam bucla pentru a citi alt bloc de caractere

jmp bucla

cleanup:

; apelam functia fclose pentru a inchide fişierul

; fclose(descriptor\_fis)

push dword [descriptor\_fis]

call [fclose]

add esp, 4

final:

; exit(0)

push dword 0

call [exit]

## Probleme propuse

A.

1. Sa se citeasca de la tastatura doua numere (in baza 10) şi sa se calculeze produsul lor. Rezultatul inmultirii se va salva in memorie in variabila "rezultat" (definita in segmentul de date).
2. Sa se citeasca de la tastatura doua numere a şi b (in baza 10) şi sa se calculeze a/b. Catul impartirii se va salva in memorie in variabila "rezultat" (definita in segmentul de date). Valorile se considera cu semn.
3. Se dau doua numere naturale a şi b (a, b: dword, definite in segmentul de date). Sa se calculeze suma lor şi sa se afiseze in urmatorul format: "<a> + <b> = <result>"  
   Exemplu: "1 + 2 = 3"  
   Valorile vor fi afisate in format decimal (baza 10) cu semn.
4. Se dau doua numere naturale a şi b (a, b: word, definite in segmentul de date). Sa se calculeze produsul lor şi sa se afiseze in urmatorul format: "<a> \* <b> = <result>"  
   Exemplu: "2 \* 4 = 8"  
   Valorile vor fi afisate in format decimal (baza 10) cu semn.
5. Se dau doua numere naturale a şi b (a, b: word, definite in segmentul de date). Sa se calculeze a/b şi sa se afiseze catul şi restul impartirii in urmatorul format: "Cat = <cat>, rest = <rest>"  
   Exemplu: pentru a=23 şi b=10 se va afisa: "Cat = 2, rest = 3"  
   Valorile vor fi afisate in format decimal (baza 10) cu semn.
6. Se dau doua numere naturale a şi b (a: dword, b: word, definite in segmentul de date). Sa se calculeze a/b şi sa se afiseze catul impartirii in urmatorul format: "<a>/<b> = <cat>"  
   Exemplu: pentru a = 200 şi b = 5 se va afisa: "200/5 = 40"  
   Valorile vor fi afisate in format decimal (baza 10) cu semn.
7. Se dau doua numere natural a şi b (a: dword, b: word, definite in segmentul de date). Sa se calculeze a/b şi sa se afiseze restul impartirii in urmatorul format: "<a> mod <b> = <rest>"  
   Exemplu: pentru a = 23 şi b = 5 se va afisa: "23 mod 5 = 3"  
   Valorile vor fi afisate in format decimal (baza 10) cu semn.
8. Se da un număr natural a (a: dword, definit in segmentul de date). Sa se citeasca un număr natural b şi sa se calculeze: a + a/b. Sa se afiseze rezultatul operatiei. Valorile vor fi afisate in format decimal (baza 10) cu semn.
9. Sa se citeasca de la tastatura doua numere a şi b (in baza 10) şi sa se calculeze: (a+b) / (a-b). Catul impartirii se va salva in memorie in variabila "rezultat" (definita in segmentul de date). Valorile se considera cu semn.
10. Sa se citeasca de la tastatura un număr in baza 10 şi sa se afiseze valoarea acelui număr in baza 16.   
    Exemplu: Se citeste: 28; se afiseaza: 1C
11. Sa se citeasca de la tastatura un număr in baza 16 şi sa se afiseze valoarea acelui număr in baza 10.  
    Exemplu: Se citeste: 1D; se afiseaza: 29
12. Se da un număr natural negativ a (a: dword). Sa se afiseze valoarea lui in baza 10 şi in baza 16, in urmatorul format: "a = <base\_10> (baza 10), a = <base\_16> (baza 16)"
13. Sa se citeasca de la tastatura doua numere a şi b (in baza 10) şi sa se calculeze: (a+b) \* (a-b). Rezultatul inmultirii se va salva in memorie in variabila "rezultat" (definita in segmentul de date).
14. Sa se citeasca de la tastatura doua numere a şi b (in baza 16) şi sa se calculeze a+b. Sa se afiseze rezultatul adunarii in baza 10.
15. Sa se citeasca de la tastatura doua numere a şi b (in baza 10) şi sa se calculeze a+b. Sa se afiseze rezultatul adunarii in baza 16.
16. Sa se citeasca de la tastatura doua numere a şi b (in baza 10). Sa se calculeze şi sa se afiseze media lor aritmetica in baza 16.
17. Sa se citeasca de la tastatura un număr in baza 10 şi sa se afiseze valoarea acelui număr in baza 16
18. Sa se citeasca de la tastatura un număr in baza 10 şi un număr in baza 16. Sa se afiseze in baza 10 numărul de biti 1 ai sumei celor doua numere citite. Exemplu:  
    a = 32 = 0010 0000b  
    b = 1Ah = 0001 1010b  
    32 + 1Ah = 0011 1010b  
    Se va afisa pe ecran valoarea 4.
19. Sa se citeasca de la tastatura un octet şi un cuvant. Sa se afiseze pe ecran daca bitii octetului citit se regasesc consecutiv printre bitii cuvantului. Exemplu:  
    a = 10 = 0000 1010b  
    b = 256 = 0000 0001 0000 0000b  
    Pe ecran se va afisa NU.  
    a = 0Ah = 0000 1010b  
    b = 6151h = 0110 0001 0101 0001b  
    Pe ecran se va afisa DA (bitii se regasesc pe pozitiile 5-12).
20. Sa se citeasca de la tastatura in baza 16 doua numere a şi b de tip dword şi sa se afiseze suma partilor low şi diferenta partilor high. Exemplu:  
    a = 00101A35h, b = 00023219h  
    suma = 4C4Eh  
    diferenta = Eh
21. Sa se citeasca de la tastatura doua numere a şi b de tip word. Sa se afiseze in baza 16 numărul c de tip dword pentru care partea low este suma celor doua numere, iar partea high este diferenta celor doua numere. Exemplu:  
    a = 574, b = 136  
    c = 01B602C6h
22. Se citesc de la tastatura doua numere a şi b. Sa se calculeze valoarea expresiei (a+b)\*k, k fiind o constanta definita in segmentul de date. Afisati valoarea expresiei (in baza 10).
23. Sa se citeasca de la tastatura un număr hexazecimal format din 2 cifre. Sa se afiseze pe ecran acest număr in baza 10, interpretat atat ca număr fara semn cat şi ca număr cu semn (pe 8 biti).
24. Se citesc de la tastatura doua numere a şi b. Sa se calculeze valoarea expresiei (a/b)\*k, k fiind o constanta definita in segmentul de date. Afisati valoarea expresiei (in baza 2).
25. Sa se citeasca de la tastatura doua numere a şi b (in baza 10) şi să se determine relaţia de ordine dintre ele (a < b, a = b sau a > b). Afisati rezultatul în următorul format: "<a> < <b>, <a> = <b> sau <a> > <b>".
26. Se citesc de la tastatura doua numere a şi b. Sa se calculeze valoarea expresiei (a-b)\*k, k fiind o constanta definita in segmentul de date. Afisati valoarea expresiei (in baza 16).
27. Se dă un sir de caractere (definit in segmentul de date). Să se citească de la tastatură un caracter, să se determine numărul de apariţii al acelui caracter în şirul dat şi să se afişeze acel caracter împreună cu numărul de apariţii al acestuia.
28. Se citesc de la tastatura numere (in baza 10) pana cand se introduce cifra 0. Determinaţi şi afişaţi cel mai mare număr dintre cele citite.
29. Se citesc de la tastatura trei numere a, m şi n (a: word, 0 <= m, n <= 15, m > n). Sa se izoleze bitii de la m-n ai lui a şi sa se afiseze numărul intreg reprezentat de acesti bitii in baza 10.
30. Se citesc de la tastatura numere (in baza 10) pana cand se introduce cifra 0. Determinaţi şi afişaţi cel mai mic număr dintre cele citite.

B.

1. Se da un fişier text. Sa se citeasca continutul fişierului, sa se contorizeze numărul de vocale şi sa se afiseze aceasta valoare. Numele fişierului text este definit in segmentul de date.
2. Se da un fişier text. Sa se citeasca continutul fişierului, sa se contorizeze numărul de consoane şi sa se afiseze aceasta valoare. Numele fişierului text este definit in segmentul de date.
3. Se da un fişier text. Sa se citeasca continutul fişierului, sa se contorizeze numărul de cifre pare şi sa se afiseze aceasta valoare. Numele fişierului text este definit in segmentul de date.
4. Se da un fişier text. Sa se citeasca continutul fişierului, sa se contorizeze numărul de cifre impare şi sa se afiseze aceasta valoare. Numele fişierului text este definit in segmentul de date.
5. Se da un fişier text. Sa se citeasca continutul fişierului, sa se contorizeze numărul de caractere speciale şi sa se afiseze aceasta valoare. Numele fişierului text este definit in segmentul de date.
6. Se da un fişier text. Sa se citeasca continutul fişierului, sa se determine cifra cu cea mai mare frecventa şi sa se afiseze acea cifra impreuna cu frecventa acesteia. Numele fişierului text este definit in segmentul de date.
7. Se da un fişier text. Sa se citeasca continutul fişierului, sa se determine litera mica (lowercase) cu cea mai mare frecventa şi sa se afiseze acea litera, impreuna cu frecventa acesteia. Numele fişierului text este definit in segmentul de date.
8. Se da un fişier text. Sa se citeasca continutul fişierului, sa se determine litera mare (uppercase) cu cea mai mare frecventa şi sa se afiseze acea litera, impreuna cu frecventa acesteia. Numele fişierului text este definit in segmentul de date.
9. Se da un fişier text. Sa se citeasca continutul fişierului, sa se determine caracterul special (diferit de litera) cu cea mai mare frecventa şi sa se afiseze acel caracter, impreuna cu frecventa acestuia. Numele fişierului text este definit in segmentul de date.
10. Sa se citeasca de la tastatura un nume de fişier şi un text. Sa se creeze un fişier cu numele dat in directorul curent şi sa se scrie textul in acel fişier. Observatii: Numele de fişier este de maxim 30 de caractere. Textul este de maxim 120 de caractere.
11. Se da un nume de fişier (definit in segmentul de date). Sa se creeze un fişier cu numele dat, apoi sa se citeasca de la tastatura cuvinte şi sa se scrie in fişier cuvintele citite pana cand se citeste de la tastatura caracterul '$'.
12. Se da un nume de fişier (definit in segmentul de date). Sa se creeze un fişier cu numele dat, apoi sa se citeasca de la tastatura numere şi sa se scrie valorile citite in fişier pana cand se citeste de la tastatura valoarea 0.
13. Se dau un nume de fişier şi un text (definite in segmentul de date). Textul contine litere mici, litere mari, cifre şi caractere speciale. Sa se transforme toate literele mici din textul dat in litere mari. Sa se creeze un fişier cu numele dat şi sa se scrie textul obtinut in fişier.
14. Se dau un nume de fişier şi un text (definite in segmentul de date). Textul contine litere mici, litere mari, cifre şi caractere speciale. Sa se transforme toate literele mari din textul dat in litere mici. Sa se creeze un fişier cu numele dat şi sa se scrie textul obtinut in fişier.
15. Se dau un nume de fişier şi un text (definite in segmentul de date). Textul contine litere mici, litere mari, cifre şi caractere speciale. Sa se inlocuiasca toate caracterele speciale din textul dat cu caracterul 'X'. Sa se creeze un fişier cu numele dat şi sa se scrie textul obtinut in fişier.
16. Se da un fişier text. Sa se citeasca continutul fişierului, sa se contorizeze numărul de litere 'y' şi 'z' şi sa se afiseze aceaste valori. Numele fişierului text este definit in segmentul de date.
17. Se da un nume de fişier (definit in segmentul de date). Sa se creeze un fişier cu numele dat, apoi sa se citeasca de la tastatura numere şi sa se scrie din valorile citite in fişier numerele divizibile cu 7, pana cand se citeste de la tastatura valoarea 0.
18. Se da un fişier text care contine litere, spatii şi puncte. Sa se citeasca continutul fişierului, sa se determine numărul de cuvinte şi sa se afiseze pe ecran aceasta valoare. (Se considera cuvant orice secventa de litere separate prin spatiu sau punct)
19. Se dau in segmentul de date un nume de fişier şi un text (poate contine orice tip de caracter). Sa se calculeze suma cifrelor din text. Sa se creeze un fişier cu numele dat şi sa se scrie suma obtinuta in fişier.
20. Se dau un nume de fişier şi un text (definite in segmentul de date). Textul contine litere mici şi spatii. Sa se inlocuiasca toate literele de pe pozitii pare cu numărul pozitiei. Sa se creeze un fişier cu numele dat şi sa se scrie textul obtinut in fişier.
21. Se dau un nume de fişier şi un text (definite in segmentul de date). Textul contine litere mici, cifre şi spatii. Sa se inlocuiasca toate cifrele de pe pozitii impare cu caracterul ‘X’. Sa se creeze un fişier cu numele dat şi sa se scrie textul obtinut in fişier.
22. Se da numele unui fişier şi un număr pe cuvant scris in memorie. Se considera numărul in reprezentarea fara semn. Sa se scrie cifrele zecimale ale acestui număr ca text in fişier, fiecare cifra pe o linie separata.
23. Se da numele unui fişier şi un număr pe cuvant scris in memorie. Sa se scrie cifrele hexazecimale ale acestui număr ca text in fişier, fiecare cifra pe o linie separata.
24. Se dau un nume de fişier şi un text (definite in segmentul de date). Textul contine litere mici, litere mari, cifre şi caractere speciale. Sa se inlocuiasca toate CIFRELE din textul dat cu caracterul 'C'. Sa se creeze un fişier cu numele dat şi sa se scrie textul obtinut prin inlocuire in fişier.
25. Se dau un nume de fişier şi un text (definite in segmentul de date). Textul contine litere mici, litere mari, cifre şi caractere speciale. Sa se inlocuiasca toate spatiile din textul dat cu caracterul 'S'. Sa se creeze un fişier cu numele dat şi sa se scrie textul obtinut prin inlocuire in fişier.
26. Se da un nume de fişier (definit in segmentul de date). Sa se creeze un fişier cu numele dat, apoi sa se citeasca de la tastatura cuvinte pana cand se citeste de la tastatura caracterul '$'. Sa se scrie in fişier doar cuvintele care contin cel putin o litera mare (uppercase).
27. Se da un fişier text. Fişierul contine numere (in baza 10) separate prin spatii. Sa se citeasca continutul acestui fişier, sa se determine minimul numerelor citite şi sa se scrie rezultatul la sfarsitul fişierului.
28. Se da un nume de fişier (definit in segmentul de date). Sa se creeze un fişier cu numele dat, apoi sa se citeasca de la tastatura cuvinte pana cand se citeste de la tastatura caracterul '$'. Sa se scrie in fişier doar cuvintele care contin cel putin o litera mica (lowercase).
29. Se da un fişier text. Fişierul contine numere (în baza 10) separate prin spatii. Sa se citeasca continutul acestui fişier, sa se determine maximul numerelor citite şi sa se scrie rezultatul la sfarsitul fişierului.
30. Se dă un nume de fişier (definit în segmentul de date). Să se creeze un fişier cu numele dat, apoi să se citească de la tastatură cuvinte până când se citeşte de la tastatură caracterul '$'. Să se scrie în fişier doar cuvintele care conţin cel puţin o cifră.