**Lab 8 – Partea II.**

**Fisiere in asamblare**

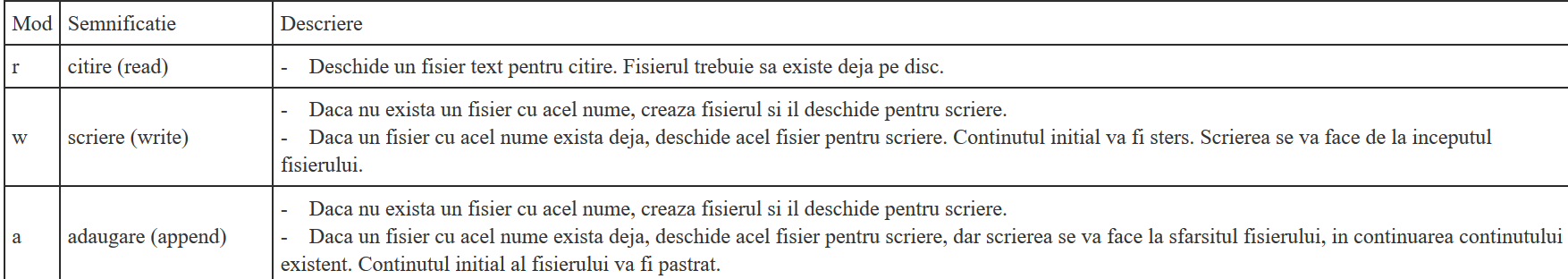
Un fisier reprezinta o secventa de octeti.

Pentru a citi dintr-un fisier sau pentru a scrie intr-un fisier, e nevoie de 3 pasi:

* Deschiderea fisierului, care poate consta in:
  + Deschiderea unui fisier existent
  + Crearea unui fisier nou
* Efectuarea operatiilor de scriere si/sau citire
* Inchiderea fisierului.

**Deschiderea unui fisier**





 Numele unui fisier trebuie sa includa si extensia (ex: nume.txt, exemplu.asm).

 Fisierele se vor crea sau deschide din directorul curent (in acelasi director in care se afla fisierul sursa asm).

Important: pentru a putea deschide un fisier existent folosind numele acestuia, fisierul trebuie sa se afle in acelasi director cu fisierul sursa .asm, altfel acesta nu va fi gasit.

**Valoarea returnata de functia fopen:**

Daca fisierul este deschis cu succes, functia fopen va completa in registrul EAX un identificator (adica un descriptor de fisier) care va fi folosit in continuare pentru a lucra cu acel fisier (pentru operatii de scriere, citire, etc.). Altfel (in caz de eroare), functia fopen va completa in registrul EAX valoarea 0.

*Alte observatii:*

* Este important sa se verifice valoarea returnata de functie in EAX (daca nu a fost eroare), inainte de a efectua alte operatii cu acel fisier.
* Daca in cadrul unui program se deschid mai multe fisiere diferite folosind functia fopen, fiecare valoare returnata de functiei trebuie salvata separat, deoarece reprezinta o valoare distincta prin care este identificat un fisier.
* Dupa finalizarea lucrului cu un fisier deschis, este important sa se si inchida acel fisier (de obicei se face la finalul programului – inainte de exit).

**Citirea dintr-un fisier**



Functia fread respecta conventia cdecl si se gaseste in msvcrt.dll .

***Argumentele (parametrii) functiei fread***

Primul argument al functiei fread reprezinta adresa **unui sir de elemente in care se vor completa** datele citite din fisier.

Al doilea argument reprezinta dimensiunea unui element care va fi citit din fisier.

Al treilea argument reprezinta numarul maxim de elemente care se vor citi din fisier.

Ultimul argument al functiei reprezinta descriptorul de fisier (identificatorul) returnat de apelul functiei fopen.

*De exemplu:*

In cazul citirii fisierelor text, primul argument al functiei fread este un sir de bytes si al doilea argument este 1 (=dimensiunea unui byte). Al treilea argument este dimensiunea sirului de bytes (numarul de elemente).

**Valoarea returnata de functia fread:**

Functia fread va completa **in registrul EAX numarul de elemente citite**. Daca acest numar este mai mic decat valoarea argumentului count, atunci fie apelul functiei fread a intampinat o eroare la citire, fie s-a ajuns la finalul fisierului.

Observatii:

* Fisierele text pot fi avea dimensiuni prea mari pentru putea citi continutul acestora cu un singur apel al functiei fread. In acest caz este nevoie de apeluri repetate ale functiei fread, pana cand intreg continutul fisierului este citit.

**Citirea dintr-un fisier folosing functia fscanf**

Spre deosebire de functia fread care este folosita cu pecadere pentru citirea sirurilor de caractere, pentru citirea numelor, in diferite baze de numeratie, se foloseste functia fscanf.

Parametrii functiei **fscanf (descriptor\_fisier, format\_citire, variabile...)**

**Exemplu: citire nr a din fisier:**

**push** **dword** a

**push** **dword** format\_citire ; format\_citire db ’%d’, 0 definit in ds

**push** **dword** **[**descriptor**] ; identificatorul de fisier din care citim**

**call** **[**fscanf**]**

**add** **esp,** 4**\***3

**Scrierea intr-un fisier**



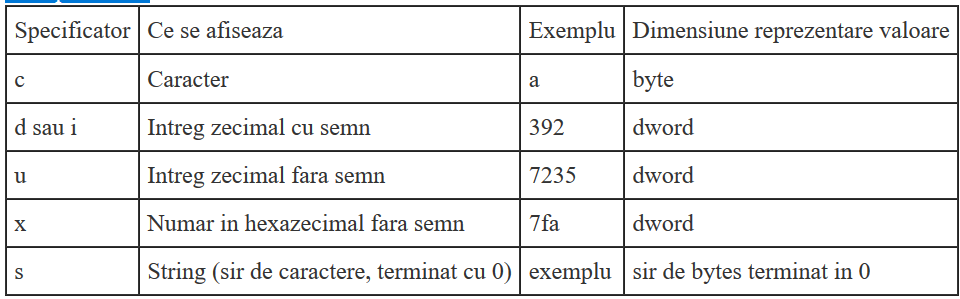
Functia fprintf respecta conventia cdecl si se gaseste in msvcrt.dll . Sintaxa functiei fprintf este asemantoare cu sintaxa functiei printf (folosita pentru afisare in consola). Diferenta este faptul ca functia fprintf are ca prim parametru identificatorul fisierului in care se va scrie textul, in plus fata de parametrii functiei printf.

Argumentele functiei fprintf

Primul argument al functiei reprezinta descriptorul de fisier (identificatorul) returnat de apelul functiei fopen.

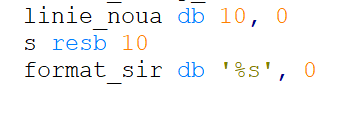
Urmatorul argument al functiei este un sir de caractere ce contine formatul afisarii, urmat de un numar de argumente (valori constante sau nume de variabile) egal cu cel specificat in cadrul formatului.

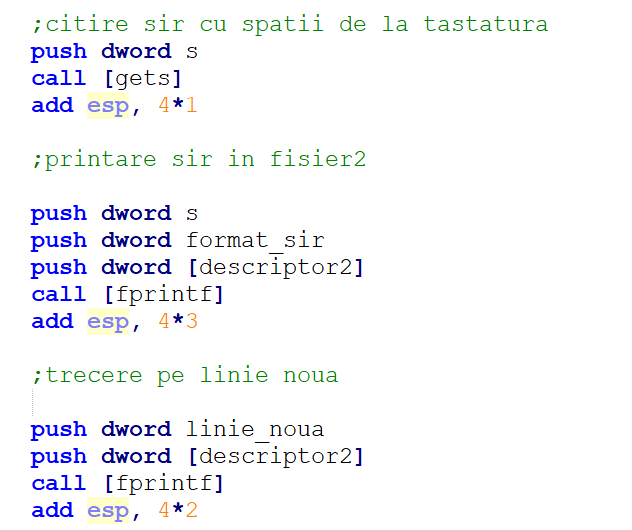
Asemenea functiei printf, sirul de caractere transmis in parametrul format poate contine anumite marcaje de formatare, ce incep cu caracterul ’%’, care vor fi inlocuite de valorile specificate in urmatoarele argumente, formatate corespunzator.



***Alte functii utile:***

***gets – permite citirea unui sir cu mai multe spatii: gets (s), s-string***





**Problema:**

; sa se citeasca 2 numere a si b din fisier1.txt.

; Sa se afiseze in fisierul 2 rezultatul operatiei a/b in fisier2.

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

extern exit**,** fopen**,** fprintf**,** fclose**,** fscanf

import exit msvcrt.dll

import fopen msvcrt.dll

import fprintf msvcrt.dll

import fscanf msvcrt.dll

import fclose msvcrt.dll

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

segment data use32 class**=**data

; ...

fisier1 db "decitit.txt"**,** 0 ; filename to be created

fisier2 db "deafisat.txt"**,**0

access\_mode2 db "w"**,** 0 ; file access mode:

access\_mode1 db "r"**,** 0 ; file access mode:

; w - creates an empty file for writing ; r - for reading

descriptor1 dd **-**1 ; variable to hold the file descriptor for fisier1

descriptor2 dd **-**1 ; ; variable to hold the file descriptor for fisier2

text db "text de afisat extra"**,** 0 ; text to be write to file

a dd 0

b dd 0

cat dd 0

restul dd 0

format\_citire db '%d'**,** 0

format\_mesaj\_afisare db ' Cele doua numere citite sunt a= %d si b=%d'**,** 0

format\_mesaj\_afisare2 db ' Rezultatul operatiei %d / %d este: catul = %d si restul = %d'**,** 0

linie\_noua db 10**,** 0

; our code starts here

segment code use32 class**=**code

start**:**

; ...

; call fopen() to open the file for reading

; fopen() will return a file descriptor in the EAX or 0 in case of error

; eax = fopen(file\_name, access\_mode)

**push** **dword** access\_mode1

**push** **dword** fisier1

**call** **[**fopen**]**

**add** **esp,** 4**\***2 ; clean-up the stack

**mov** **[**descriptor1**],** **eax** ; store the file descriptor returned by fopen

; check if fopen() has successfully created the file (EAX != 0)

**cmp** **eax,** 0

**je** final

; daca a reusit citirea:

;citire din fiser1 folosind fscanf

; citire primul nr a din fisier

**push** **dword** a

**push** **dword** format\_citire

**push** **dword** **[**descriptor1**]**

**call** **[**fscanf**]**

**add** **esp,** 4**\***3

; citire al doilea nr din fisier: b

**push** **dword** b

**push** **dword** format\_citire

**push** **dword** **[**descriptor1**]**

**call** **[**fscanf**]**

**add** **esp,** 4**\***3

; call fopen() to create the file

; fopen() will return a file descriptor in the EAX or 0 in case of error

; eax = fopen(file\_name, access\_mode)

**push** **dword** access\_mode2

**push** **dword** fisier2

**call** **[**fopen**]**

**add** **esp,** 4**\***2 ; clean-up the stack

**mov** **[**descriptor2**],** **eax** ; store the file descriptor returned by fopen

; check if fopen() has successfully created the file (EAX != 0)

**cmp** **eax,** 0

**je** final

;daca e totul ok, efectuam calculele

**mov** **eax,** **[**a**]**

**cdq**

**idiv** **dword[**b**]**

**mov** **[**cat**],** **eax**

**mov** **[**restul**],** **edx**

;afisare mesaj in fisier2

**push** **dword** **[**b**]**

**push** **dword** **[**a**]**

**push** **dword** format\_mesaj\_afisare

**push** **dword** **[**descriptor2**]**

**call** **[**fprintf**]**

**add** **esp,** 4**\***4

;trecere pe linie noua

**push** **dword** linie\_noua

**push** **dword** **[**descriptor2**]**

**call** **[**fprintf**]**

**add** **esp,** 4**\***2

; write the text to file using fprintf()

; fprintf(file\_descriptor, text)

**push** **dword** **[**restul**]**

**push** **dword** **[**cat**]**

**push** **dword** **[**b**]**

**push** **dword** **[**a**]**

**push** **dword** format\_mesaj\_afisare2

**push** **dword** **[**descriptor2**]**

**call** **[**fprintf**]**

**add** **esp,** 4**\***6

; call fclose() to close the file

; fclose(file\_descriptor)

**push** **dword** **[**descriptor2**]**

**call** **[**fclose**]**

**add** **esp,** 4

**push** **dword** **[**descriptor1**]**

**call** **[**fclose**]**

**add** **esp,** 4

final**:**

; exit(0)

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

**call** **[**exit**]** ; call exit to terminate the program