Lab 12 – Lucrul cu fisiere în limbaj de asamblare pe 32 biți

Ce este un fisier de tip text?

Un fișier text este un tip de fișier în care datele sunt stocate ca o secvență de caractere, într-o codificare predefinită, în general folosind codificarea ASCII (American Standard Code for Information Interchange); tipul de date este considerat ca fiind textul neformatat. Într-un astfel de fisier, fiecare rând de text este delimitat de unul sau mai multe ***caractere de sfârșit de rând*** (caractere EOL - End Of Line). Acestea diferă în funcție de sistemul de operare folosit pentru crearea și editarea fișierului:

- Windows folosește două caractere de control (ASCII 13 urmat de ASCII 10, sau, mai scurt CR (Carriage Return)+LF Line Feed – asa cum se folosea la masinile de scris)
- sistemele tip UNIX (incluzând aici și Linux și Mac OS X) folosesc numai caracterul LF,
- sistemele Mac OS pre-Unix (versiunile 9 sau mai vechi), folosesc doar caracterul CR

Un fisier reprezintă de fapt o secvență de octeți.

Pentru a citi dintr-un fisier sau pentru a scrie intr-un fisier, în general se folosesc 3 pasi:

I. **Deschiderea fisierului**, care poate consta in:

Deschiderea unui fisier existent sau ***Crearea unui fisier nou***

II. **Efectuarea operatiilor** de scriere si/sau citire

III. **Închiderea fisierului**.

Funcții des folosite în lucrul cu fisiere:

În ***limbajul C*** există o serie de funcții specializate pentru lucrul cu fisiere. În C, pentru a se putea folosi aceste funcții, este necesară includerea fisierului header stdio.h, dar în ***limbajul de asamblare***, se va folosi importarea acestor funcții din librăria **MSVCRT.dll,** acestea fiind declarate ca fiind externe cu directiva **extern.**

Principalele funcții folosite în lucrul cu fisiere sunt următoarele:

Nume functie	*Descriere functie*
fopen	Deschide un fisier
fclose	Închide un fisier
fputch	Scrie un caracter într-un fisier
fgetch	Citeste un caracter dintr-un fisier
fseek	Caută un anumit octet într-un fisier
fprintf	Echivalentul lui printf (), folosit pentru un fisier
fscanf	Echivalentul lui scanf (), folosit pentru un fisier
fread	
fwrite	
feof	Returnează o valoare diferită de 0 dacă se ajunge la sfârsitul fisierului
ferror	Returnează o valoare diferită de 0 dacă apare o eroare
ftell	Returnează poziția cursorului într-un fisier

Pentru a se putea utiliza un fisier, este necesară folosirea unui pointer, care să facă legătura între fisierul respectiv si sistemul I/O.

Pointer-ul respectiv este un pointer către anumite informații despre fisierul respectiv – nume, starea si poziția in care se află pointerul de fisier, dimensiune, etc. Acest pointer este în general reprezentat sub forma:

```
FILE * fisier;
```

Deschiderea fisierelor

Deschiderea unui fisier se realizează cu funcția **fopen**, care are următorul prototip în C:

FILE *fopen (char *nume_fisier, char *mod_acces)

Primul parametru (nume_fisier) reprezintă numele fisierului care urmează a fi deschis. Al doilea parametru (mod_acces) este un sir de caractere care indică modul de accesare al fisierului. Valorile posibile pentru acest parametru sunt următoarele:

Mod_acces	Semnificație	Descriere
r	Citire (read)	doar citire; dacă fisierul nu există se semnalează eroare
w	Scriere (write)	doar scriere; dacă fisierul nu există, va fi creat, iar dacă există, va fi suprascris -Daca nu exista un fisier cu acel nume, **creaza fisierul si il deschide pt scriere**. -Daca un fisier cu acel nume exista deja, deschide acel fisier pentru scriere. **Continutul initial va fi sters**. Scrierea se va face de la inceputul fisierului. **Exemplu:** am avut in fisier: Ana si Mihai au multe mingi. • am folosit w pt scriere: "Ion este bun" => "Ion este bun"
a	Adăugare (append)	doar scriere; dacă fisierul nu există va fi creat, iar dacă există se va adauga la sfârsitul lui ⇨ Daca un fisier cu acel nume exista deja, deschide acel fisier pentru scriere, dar scrierea se va face la sfarsitul fisierului, in continuarea continutului existent. Continutul initial al fisierului va fi pastrat. -Daca nu exista un fisier cu acel nume, **creaza fisierul si il deschide pentru scrier**e. -Daca un fisier cu acel nume exista deja, deschide acel fisier pentru scriere, dar **scrierea se va face la sfarsitul fisierului**, in continuarea continutului existent. **Continutul initial al fisierului va fi pastrat.** **Exemplu:** am avut in fisier: Ana si Mihai au multe mingi. • am folosit a pt scriere: "Ion este bun" => "Ana si Mihai au multe mingi.Ion este bun"
r+	Citire si scriere într-un fisier existent	citire si scriere; dacă fisierul nu există se semnalează eroare ⇨ Deschide un fisier text pentru citire si scriere. Fisierul trebuie sa existe deja pe disc. -Deschide un fisier text pentru citire si scriere. **Fisierul trebuie sa existe deja pe disc. In functie de lungimea continutului initial, vom vedea ca suprascrie (pastreaza textul pentru citire, dar pozitioneaza pointerul pentru scriere la inceput)**

		Exemplu: am avut in fisier: Maria are pere. Ana are mere. • am folosit r+ pt scriere: “Mihai are o minge.” => “Mihai are o minge. are mere.”
w+	Citire si scriere	citire si scriere; dacă fisierul nu există va fi creat, iar dacă fisierul există va fi suprascris ⇨ Daca nu exista un fisier cu acel nume, creaza fisierul si il deschide pentru citire si scriere. ⇨ Daca un fisier cu acel nume exista deja, deschide acel fisier pentru citire si scriere. Continutul initial va fi sters. Scrierea se va face de la inceputul fisierului. -Daca nu exista un fisier cu acel nume, creaza fisierul si il deschide pentru citire si scriere. -Daca un fisier cu acel nume exista deja, deschide acel fisier pt citire si scriere. **Continutul initial va fi sters.** Scrierea se va face de la inceputul fisierului. **Exemplu:** am avut in fisier: Ana si Mihai au multe mingi. • am folosit w+ pt scriere: “Ion este bun” => “Ion este bun”
a+	Citire si adăugare	citire si scriere; dacă fisierul nu există va fi creat, iar dacă fisierul există se va adăuga la sfârsitul său ⇨ Daca nu exista un fisier cu acel nume, creaza fisierul si il deschide pentru citire si scriere. ⇨ Daca un fisier cu acel nume exista deja, deschide acel fisier pentru citire si scriere. Continutul initial al fisierului va fi pastrat. Citirea se va face de la inceputul fisierului. Scrierea se va face in continuarea continutului existent.

Observatii:

- 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.
- Operatiile de scriere nu vor reusi pentru fisiere deschise doar cu drepturi de citire (ex: ”r”). Operatiile de citire nu vor reusi pentru fisiere deschise doar cu drepturi de scriere sau adaugare (ex: ”w”, ”a”)

Scrierea într-un fişier

Pentru a scrie un text intr-un fisier se foloseste functia **fprintf**.

Sintaxa functiei **fprintf** in limbaj de programare de nivel inalt este:

```
int fprintf(FILE * stream, const char * format, <variabila_1>, <constanta_2>, <...>)
```

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.

Specificator	Ce se afiseaza	Exemplu	Dimensiune reprez. valoare
c	Caracter	a	byte
d sau i	Intreg zecimal cu semn	392	dword
u	Intreg zecimal fara semn	7235	dword
x	Numar in hexa fara semn	7fa	dword
s	String (sir de caractere, terminat cu 0)	exemplu	sir de bytes terminat in 0

Citirea dintr-un fişier:

Pentru a citi un text dintr-un fisier se foloseste functia **fread**.

Sintaxa functiei **fread** in limbaj de programare de nivel inalt este:

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

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

Argumentele 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*. **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. In sectiunea „Exemple" vom prezenta un program care exemplifica acest scenariu. Pentru a verifica daca s-a ajuns la finalul fisierului cu operatia de citire se poate verifica daca valoarea returnata de fread este 0.

Închiderea unui fişier deschis

Dupa finalizarea lucrului cu un fisier deschis, acesta trebuie inchis. Acest pas nu trebuie sa lipseasca dintr-un program care a deschis fisiere. Pentru inchiderea unui fisier se foloseste functia **fclose**.

Sintaxa functiei **fclose** in limbaj de programare de nivel inalt este:

```
int fclose(FILE * descriptor)
```

Functia fclose respecta conventia *cdecl* si se gaseste in *msvcrt.dll* .

Argumentul functiei fclose: Argumentul functiei fclose este descriptorul de fisier (identificatorul) returnat la apelul functiei *fopen*.

Exemplul 1:

Exemplu de program in C:

crearea (deschiderea) unui fisier si setarea atributului "w" (pentru a se putea scrie în fisierul respectiv):

```
#include <stdio.h>
int main () {
FILE *fisier;
fisier = fopen("myFile.txt", "w");
if (fisier == NULL) { printf ("Fisierul nu poate fi creat."); }
else { printf ("Fisierul a fost creat cu succes");
fprintf(fisier, "%s", "Ana are mere");
fclose(fisier); }
return(0);
}
```

-> utilizatorului i se confirmă crearea fisierului dorit, iar în caz contrar (protecţie la scriere, disc plin etc.) se va afisa un mesaj.

Exemplul 2:

Un exemplu de program in ASM pe 32 biți: scrierea unui text intr-un fisier

```
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

segment data use32 class=data
nume_fisier db "input.txt", 0 ; numele fisierului care va fi creat
mod_acces db "w", 0           ; modul de deschidere al fisierului -
                              ; w - pentru scriere. daca fiserul nu exista, se va crea
text db "Ana are mere.", 0    ; textul care va fi scris in fisier
```

```
descriptor_fis dd -1            ; variabila in care vom salva descriptorul fisierului - necesar pentru a putea face referire la fisier

segment code use32 class=code
start:
; apelam fopen pentru a crea 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                    ; 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 fisierul (daca EAX != 0)
cmp eax, 0
je final

; scriem textul in fisierul 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 fisierul
; fclose(descriptor_fis)
push dword [descriptor_fis]
call [fclose]
add esp, 4


final:
; exit(0)
push dword 0
call [exit]
```

Exemplul 3:

EXEMPLU:

Citirea unui text scurt (max 100 caractere) dintr-un fisier

; Codul de mai jos va deschide un fisier numit "ana.txt" din directorul curent si va citi un text de maxim 100 de caractere din acel fisier.
; Programul va folosi functia fopen pentru deschiderea fisierului, functia fread pentru citirea din fisier si functia fclose pentru inchiderea fisierului deschis.
; Deoarece in apelul functiei fopen programul foloseste modul de acces "r", daca un fisier cu numele dat nu exista in directorul curent, apelul functiei fopen nu va reusi (eroare).
Detalii despre modurile de acces sunt prezentate in sectiunea "Suport teoretic".

```
bits 32
global start
extern exit, fopen, fread, fclose
import exit msvcrt.dll
import fopen msvcrt.dll
import fread msvcrt.dll
import fclose msvcrt.dll


segment data use32 class=data
    nume_fisier db "ana.txt", 0  ; numele fisierului care va fi deschis
    mod_acces db "r", 0          ; modul de deschidere a fisierului - r -pt scriere. fisierul trebuie sa existe
    descriptor_fis dd -1         ;aici vom salva descriptorul fisierului - necesar pentru a putea face referire la fisier
    len equ 100                  ; numarul maxim de elemente citite din fisier.
    text times len db 0          ; sirul in care se va citi textul din fisier


segment code use32 class=code
    start:
        ; apelam fopen pt 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                ; eliberam parametrii de pe stiva


        mov [descriptor_fis], eax   ; salvam valoarea returnata de fopen in variabila descriptor_fis

        cmp eax, 0          ; verificam daca functia fopen a creat cu succes fisierul (daca EAX != 0)
        je final

        ; citim textul in fisierul 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 numarul de caractere citite din fisier

    ; apelam functia fclose pentru a inchide fisierul
    ; fclose(descriptor_fis)
    push dword [descriptor_fis]
    call [fclose]
    add esp, 4

  final:        ; exit(0)
    push    dword 0
    call    [exit]
```

Exemplul 4:

Citirea unui text scurt (max 100 caractere) dintr-un fisier – cu afisarea textului

; Codul de mai jos va deschide un fisier numit "ana.txt" din directorul curent, va citi un text scurt din acel fisier, apoi va afisa in consola numarul de caractere citite si textul citit din fisier.
; Programul va folosi functia fopen pentru deschiderea fisierului, functia fread pentru citirea din fisier si functia fclose pentru inchiderea fisierului creat.
; Deoarece in apelul functiei fopen programul foloseste modul de acces "r", daca un fisier numele dat nu exista in directorul curent, apelul functiei fopen nu va reusi (eroare). Detalii despre modurile de acces sunt prezentate in sectiunea "Suport teoretic".

; In acest program sirul de caractere in care se va citi textul din fisier trebuie sa aiba o lungime cu 1 mai mare decat numarul maxim de elemente care vor fi citite din fisier 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 fisierul ar contine mai mult de <len> caractere si dimensiunea sirului destinatie era exact <len>, intregul sir ar fi fost completat cu valori citite din fisier, astfel sirul nu se mai termina cu valoarea 0.

```
bits 32
global start
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


segment data use32 class=data
    nume_fisier db "ana.txt", 0  ; numele fisierului care va fi creat
    mod_acces db "r", 0          ; modul de deschidere a fisierului -  r - pentru scriere. fisierul trebuie sa existe
    len equ 100                  ; numarul maxim de elemente citite din fisier.
    text times (len+1) db 0      ; sirul in care se va citi textul din fisier (dimensiune len+1 explicata mai sus)
    descriptor_fis dd -1         ; aici vom salva descriptorul fisierului - necesar pentru a putea face referire la fisier
    format db "Am citit %d caractere din fisier. Textul este: %s", 0  ; formatul - utilizat pentru afisarea textului citit din fisier   -  %s reprezinta un sir de caractere


segment code use32 class=code
```

```
start:
    ; apelam fopen pt 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                ; eliberam parametrii de pe stiva


    mov [descriptor_fis], eax   ; salvam valoarea returnata de fopen in variabila descriptor_fis

    cmp eax, 0     ; verificam daca functia fopen a creat cu succes fisierul (daca EAX != 0)
    je final

    ; citim textul in fisierul 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 numarul de caractere citite din fisier

    ; afisam numarul de caractere citite si 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 fisierul
    ; fclose(descriptor_fis)
    push dword [descriptor_fis]
    call [fclose]
    add esp, 4

  final:        ; exit(0)
    push    dword 0
    call    [exit]
```

Exemplul 5:

Citirea intregului text dintr-un fisier (in etape) - cu afisarea continutului fisierului (prin curatarea bufferului)

; Codul de mai jos va deschide un fisier numit "input.txt" din directorul curent si va citi intregul text din acel fisier, in etape, cate 100 de caractere intr-o etapa.
; Deoarece un fisier text poate fi foarte lung, nu este intotdeauna posibil sa citim fisierul intr-o singura etapa pentru ca nu putem defini un sir de caractere suficient de lung pentru intregul text din fisier. De aceea, prelucrarea fisierelor text in etape este necesara.

; Programul va folosi functia fopen pentru deschiderea fisierului, functia fread pentru citirea din fisier si functia fclose pentru inchiderea fisierului creat. Deoarece in apelul functiei fopen programul foloseste modul de acces "r", daca un fisier numele dat nu exista in directorul curent, apelul functiei fopen nu va reusi (eroare). Detalii despre modurile de acces sunt prezentate in sectiunea "Suport teoretic".

```
bits 32
global start
extern exit, fopen, fclose, fread, printf
import exit msvcrt.dll
import fopen msvcrt.dll
import fread msvcrt.dll
import fclose msvcrt.dll
import printf msvcrt.dll


segment data use32 class=data
    nume_fisier db "input.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            ; aici 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 citit din fisier in etapa curenta
    len equ 100                     ; numarul maxim de elemente citite din fisier intr-o etapa
    buffer times len+1 db 0                 ; sirul in care se va citi textul din fisier
format db 10,13,"Am citit %d caractere din fisier. Textul este: ",10,13,"%s", 0  ;


segment code use32 class=code
    start:
        ; apelam fopen pt 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


        cmp eax, 0              ; verificam daca functia fopen a creat cu succes fisierul (daca 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 fisierul 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 = numar de caractere / bytes citite
        cmp eax, 0          ; daca numarul de caractere citite este 0, am terminat de parcurs fisierul
        je cleanup


        mov [nr_car_citite], eax        ; salvam numarul de caractere citie

        ; instructiunile pentru procesarea caracterelor citite in aceasta etapa incep aici
        ; …
;printf "Am citit %d caractere din fisier. Textul este: %s", 0
        push dword buffer
        push dword EAX
        push dword format
        call [printf]
        add ESP, 4*3


; curat bufferul
        mov EDI,0
        mov ECX, len
        et:
          mov byte [buffer + EDI],0
          inc EDI
        loop et
          ; reluam bucla pentru a citi alt bloc de caractere
          jmp bucla
cleanup:
        ; apelam functia fclose pentru a inchide fisierul
        ; fclose(descriptor_fis)
        push dword [descriptor_fis]
        call [fclose]
        add esp, 4

      final:        ; exit(0)
        push    dword 0
        call    [exit]
```