

# Tema laborator 8-9

Pop Mihai-Daniel, Grupa 215/2

## Problema nr. 4 - Apeluri de functii sistem

4. Se dau doua numere naturale a si b (a, b: word, definite in segmentul de date). Sa se calculeze produsul lor si sa se afiseze in urmatorul format: "<a> \* <b> = <result>"

Exemplu: "2 \* 4 = 8"

Valorile vor fi afisate in format decimal (baza 10) cu semn.

```
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 dw 0          ;definim variabila word a
    b dw 0          ;definim variabila word b
    format db "%d", 0 ; %d <=> un numar decimal (baza 10)
    format1 db "a = ", 0 ; mesaj pentru selectarea numarului a
    format2 db "b = ", 0 ; mesaj pentru selectarea numarului b
    message2 db "Produsul numerelor este: %d * %d = %d", 0 ; mesaj pentru afisarea produsului numerelor a si b

segment code use32 class=code
start:

    ; vom apela scanf(format, n) => se va citi un numar in variabila n
    ; punem parametrii pe stiva de la dreapta la stanga

    push dword format1 ; ! pe stiva se pune adresa string-ului, nu valoarea
    call [printf]       ; apelam functia printf pentru afisare
    add esp, 4*1        ; eliberam parametrii de pe stiva; 4 = dimensiunea unui dword; 1 = nr de parametri

    push dword a        ; punem pe stiva adresa variabilei a
    push dword format    ; punem pe stiva adresa string-ului
    call [scanf]        ; apelam functia scanf pentru citire
    add esp, 4*2        ; eliberam parametrii de pe stiva; 4 = dimensiunea unui dword; 2 = nr de parametri

    push dword format2 ; ! pe stiva se pune adresa string-ului, nu valoarea
    call [printf]       ; apelam functia printf pentru afisare
    add esp, 4*1        ; eliberam parametrii de pe stiva; 4 = dimensiunea unui dword; 1 = nr de parametri

    push dword b        ; punem pe stiva adresa variabilei a
    push dword format    ; punem pe stiva adresa string-ului
    call [scanf]        ; apelam functia scanf pentru citire
    add esp, 4*2        ; eliberam parametrii de pe stiva; 4 = dimensiunea unui dword; 2 = nr de parametri

    mov ax, word[a]     ; se pune in ax word-ul a
    imul word[b]        ; se face inmultirea cu semn a variabilelor a si b

    push dx             ; se pune in stiva partea high din rezultat
    push ax             ; se pune in stiva partea low din rezultat
    pop ebx             ; se ia de pe stiva intr-un dublu cuvant rezultatul din dx:ax

    mov ax,[a]          ; punem in ax valoarea variabilei a
    cwde               ; convertire cu semn de la cuvant la dublucuvant
    mov ecx, eax        ; salvam in ecx rezultatul
```

```

mov ax, [b]      ;punem in ax valoarea variabilei b
cwde             ;convertire cu semn de la cuvnt la dublucuvnt

push dword ebx   ;punem pe stiva valoarea rezultatului inmultirii
push dword eax   ;punem pe stiva valoarea variabilei b
push dword ecx   ;punem pe stiva valoarea variabilei a
push dword message2;punem pe stiva adresa mesajului de tip string
call [printf]    ;apelam functia printf pentru afisare
add esp,4*4      ;eliberam parametrii de pe stiva; 4 = dimensiunea unui dword; 4 = nr de parametri

;exit(0)
push dword 0     ;punem pe stiva parametrul pentru exit
call [exit]      ;apelam exit pentru a incheia programul

```

The screenshot shows an assembly editor window with the following code:

```

14 import scanf msvcrt.dll
15
16 segment data use32 class=data
17     a dw 0 ;definim variabila word a
18     b dw 0 ;definim variabila word b
19     format db "%d", 0 ; %d <=> un numar decimal (baza 10)
20     format1 db "a = ", 0 ;mesaj pentru selectarea numarului a
21     format2 db "b = ", 0 ;mesaj pentru selectarea numarului b
22     message2 db "Produsul numerelor este: %d * %d = %d"
23
24 segment code use32 class=code
25 start:
26     ; vom apela scanf(format, n) => se va citi
27     ; punem parametrii pe stiva de la dreapta
28
29     push dword format1 ; ! pe stiva se pune
30     call [printf] ; apelam functia printf
31     add esp, 4*1 ; eliberam parametrii
32
33     push dword a ;punem pe stiva adresa
34     push dword format ;punem pe stiva adresa
35     call [scanf] ; apelam functia scanf
36     add esp, 4 * 2 ; eliberam parametrii
37
38     push dword format2 ; ! pe stiva se pune
39     call [printf] ; apelam functia printf
40     add esp, 4*1 ; eliberam parametrii
41
42     push dword b ;punem pe stiva adresa
43     push dword format ;punem pe stiva adresa
44

```

The console window shows the output of the program:

```

a = 12
b = -12
Produsul numerelor este: 12 * -12 = -144
Press any key to continue . . .

```

## Problema nr. 27 - Apeluri de functii sistem

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 apariții al acelui caracter în șirul dat și să se afișeze acel caracter împreună cu numărul de apariții al acestuia.

```

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
    sir db "alabalaportocala",0 ;declararea sirului sir
    len equ $-sir ;lungimea sirului sir
    contor dd 0 ;contorul in care retinem numarul de aparitii a caracterului citit
    caracter db 0 ;caracterul citit de la tastatură
    format db "Dati caracterul: ",0;formatul pentru citire
    format_c db "%c",0 ;formatul unui caractere de tip char
    format_nr db "%d",0 ;formatul pentru un numar in baza 10
    mesaj db "Caracterul %c apare in text de %d ori",0;mesajul care se afiseaza pe ecran la finalul problemei

```

```

segment code use32 class=code
start:

```

```

push format      ; ! pe stiva se pune adresa string-ului, nu valoarea
call [printf]    ; apelam functia printf pentru afisare
add esp, 4       ; eliberam parametrii de pe stiva; 4 = dimensiunea unui dword; 1 = nr de parametri

push caracter     ; punem pe stiva adresa caracterului c
push format_c     ; punem pe stiva adresa string-ului
call [scanf]      ; apelam functia scanf pentru citire
add esp, 4*2      ; eliberam parametrii de pe stiva; 4 = dimensiunea unui dword; 2 = nr de parametri

mov esi, sir      ; punem in registrul esi adresa sirului sir
mov ecx, len      ; ecx = lungimea sirului sir
mov ebx, dword[contor] ; initializam un registru cu 0, contorul

repetă:          ; inceputul buclei
    lodsb        ; incarcarea in memorie/al a unui byte
    cmp al,[caracter] ; comparam byte-ul incarcat in al cu valoarea caracterului
    jne altfel    ; daca nu sunt egale trece la urmatorul element din sir
    inc ebx      ; daca caracterul apare in sir, incrementam registrul ebx

    altfel:      ; eticheta la care sare daca caracterul nu este egal cu pozitia curenta din sir
loop repetă      ; reluarea buclei cat timp ecx != 0

push ebx          ; punem pe stiva adresa registrului ebx
push dword[caracter] ; punem pe stiva valoarea din variabila caracter(caracterul care trebuia verificat)
push mesaj        ; pe stiva se pune adresa mesajului
call [printf]     ; apelam functia printf pentru afisare
add esp, 4*3      ; eliberam parametrii de pe stiva; 4 = dimensiunea unui dword; 3 = nr de parametri

push dword 0      ; punem pe stiva parametrul pentru exit
call [exit]       ; apelam exit pentru a incheia programul

```

```

9  import printf msvcrt.dll
10 import scanf msvcrt.dll
11
12 segment data use32 class=data
13     sir db "alabalaportocala",0 ;declararea sirului sir
14     len equ $-sir
15     contor dd 0
16     caracter db 0
17     format db "Dati caracterul: a"
18     format_c db "%c"
19     format_nr db "%d"
20     mesaj db "Caracterul a apare in text de 6 ori"
21
22 segment code use32 class=code
23     start:
24
25         push format
26         call [printf]
27         add esp, 4
28
29         push caracter
30         push format_c
31         call [scanf]
32         add esp, 4*2
33
34         mov esi, sir
35         mov ecx, len
36         mov ebx, dword[contor]
37
38         repeta:
39             lodsb

```

## Problema nr. 4 – Operatii cu fisiere

4. Se da un fisier text. Sa se citeasca continutul fisierului, sa se contorizeze numarul de cifre impare si sa se afiseze aceasta valoare. Numele fisierului text este definit in segmentul de date.

;4.Se da un fisier text. Sa se citeasca continutul fisierului, sa se contorizeze numarul de cifre impare si sa se afiseze aceasta valoare. Numele fisierului text este definit in segmentul de date

```

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

segment data use32 class=data
    nume_fisier db "lab8.txt",0 ; definim numele fisierului din care citim
    mod_acces db "r",0 ; READ, daca fisierul nu exista se va crea
    descriptor_fis dd -1 ; variabila in care vom salva descriptorul fisierului - necesar pentru a putea face referire la fisier
    mesaj db "Numarul de cifre impare din fisier este: %d",0; mesajul care trebuie afisat impreuna cu rezultatul
    len equ 100 ; definirea unei constante care reprezinta lungimea maxima a sirului
    text times len+1 db 0 ; alocarea in memorie a unui numar 'len+1' de 0-uri
    numarCaractere dd 0 ; contor in care retinem lungimea sirului
    mesaj2 db 10,13,"numarul de caractere este: %d",0; mesaj pentru numarul de caractere
    mesaj3 db 10,13,"numarul de cifre impare din fisier este: %d",0; mesaj pentru numarul cifrelor pare din fisier
    contor dd 0 ; contor care retine cate cifre impare exista in text
segment code use32 class=code
    start:
        push dword mod_acces ;punem pe stiva adresa modului de acces cu care vrem sa lucram (read)
        push dword nume_fisier ;se pune pe stiva adresa numelui fisierului
        call [fopen] ;apelarea functiei de deschidere a fisierului
        add esp,4*2 ;eliberam parametrii de pe stiva; 4 = dimensiunea unui dword; 2 = nr de parametri

        ; verificam daca functia fopen a creat cu succes fisierul (daca EAX != 0)
        mov [descriptor_fis], eax ; salvam valoarea returnata de fopen in variabila descriptor_fis
        cmp eax, 0 ; se compara valoarea din ecx cu 0
        je final ; daca e 0 sare la finalul programului

    repeta: ;eticheta de inceput
        ; 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

        cmp eax,0 ;daca nu s-a citit nimic se sare la final
        je gata
        add [numarCaractere],eax;in variabila numarCaractere retinem lungimea textului citit

    jmp repeta ;saltul la eticheta repeta

    gata: ;iesirea din functia de citire

        push dword text ;se pune pe stiva adresa textului citit
        call [printf] ;apelarea functiei de afisare pe ecran
        add esp, 4*1 ; eliberam parametrii de pe stiva; 4 = dimensiunea unui dword; 1 = nr de parametri

        push dword[numarCaractere] ;se pune pe stiva, iar apoi afisata pe ecran valoarea in care retinem lungimea textului citit
        push dword mesaj2
        call [printf]
        add esp, 4*2

        mov esi, text ;punem in registrul esi adresa textului citit
        mov ecx, dword[numarCaractere];punem in ecx lungimea textului
        jecxz final ;daca lungimea este 0 se sare la final

```

```

repet1:      ;parcurgerea textului
lodsb      ;se incarca in memorie/al un byte
cmp al,'0'   ;se verifica daca e cifra
jl next
cmp al,'9'
jg next
test al,01h  ;se verifica paritatea cifrei
je next ;      !!!!!!!! vream explicatii
add dword[contor],1 ;daca e impar crestem valoarea contorului cu 1

next:      ;eticheta la care se sare daca caracterul incarcata in memorie nu e cifra
loop repet1 ;repetarea buclei

push dword[contor] ;se pune pe stiva valoarea contorului
push dword mesaj3 ;se pune pe stiva adresa mesajului 3 pentru afisare
call [printf] ;se apeleaza functia de afisare pe ecran
add esp, 4*2 ;eliberam parametrii de pe stiva; 4 = dimensiunea unui dword; 2 = nr de parametri

push dword [descriptor_fis] ;punem pe stiva valoarea descriptorului pentru a putea inchide fisierul
call [fclose]
add esp, 4

final:      ;eticheta la care se sare in caz in care fisierul este gol

push dword 0 ;punem pe stiva parametrul pentru exit
call [exit] ;apelam exit pentru a incheia programul

```

The screenshot shows a Windows XP desktop. On the left, a Notepad++ window displays assembly code for a program that reads a file and calculates statistics. On the right, a Windows Command Prompt window shows the program's output.

**Assembly Code (Notepad++):**

```

16 import fread, fclose, printf, exit
17
18 segment data use32 class=data
19     nume_fisier db "lab8.txt",0
20     mod_acces db "r",0
21     descriptor_fis dd -1
22     mesaj db "Numarul de cifre impare din fisier este: ",0
23     len equ 100
24     text times len+1 db 0
25     numarCaractere dd 0
26     mesaj2 db 10,13,"numarul de caractere este: ",0
27     mesaj3 db 10,13,"numarul de cifre impare din fisier este: ",0
28     contor dd 0
29 segment code use32 class=code
30     start:
31         push dword mod_acces
32         push dword nume_fisier
33         call [fopen]
34         add esp, 4*2
35
36         ; verificam daca functia f
37         mov [descriptor_fis], eax
38         cmp eax, 0
39         je final
40
41     repet1:
42         ; citim textul in fisier
43         ; eax = fread(text, 1,
44         push dword [descriptor_fis]
45         push dword len
46         push dword 1
47         push dword text
48         call [fread]
49         add esp, 4*4
50
51     final:
52         push dword 0
53         call [exit]

```

**Command Prompt Output:**

```

C:\WINDOWS\system32\cmd.exe
Acesta este un document text de proba.
numarul de caractere este: 47
numarul de cifre impare din fisier este: 5
Press any key to continue . . .

```

## Problema nr. 27 – Operatii cu fisiere

27. Se da un fisier text. Fisierul contine numere (in baza 10) separate prin spatii. Sa se citeasca continutul acestui fisier, sa se determine minimul numerelor citite si sa se scrie rezultatul la sfarsitul fisierului.

;27.Se da un fisier text. Fisierul contine numere (in baza 10) separate prin spatii. Sa se citeasca continutul acestui fisier, sa se determine minimul numerelor citite si sa se scrie rezultatul la sfarsitul fisierului.

bits 32

global start

extern exit, fopen, fprintf, fclose, fread

extern scanf

import scanf msvcrt.dll

extern printf

import printf msvcrt.dll

import exit msvcrt.dll

import fopen msvcrt.dll

import fprintf msvcrt.dll

import fclose msvcrt.dll

import fread msvcrt.dll

segment data use32 class=data

```
    fisier db "file27.txt", 0    ; definim numele fisierului din care citim
    acces_mode db "a+", 0      ; APPEND+, modul in care lucram cu fisierul
    descriptor dd 0             ; variabila in care vom salva descriptorul fisierului - necesar pentru a putea face referire la fisier
    afisare db "Numarul minim din fisier este %d", 0; mesajul care trebuie afisat impreuna cu rezultatul
    minim dd 255               ; definim o variabila in care retinem minimul
    curent db 0                ; definim o variabila in care retinem numarul curent
    sir resb 100               ; rezervam in memorie 100 byte-uri pentru sir
    len equ 100                ; definim o constanta cu valoarea 100/lungimea maxima a unui text citit
    ten db 10                  ; definim o variabila 10 cu care cream numere in baza 10
```

segment code use32 class=code

start:

```
    push dword acces_mode      ; punem pe stiva adresa modului de acces cu care vrem sa lucram (append)
    push dword fisier          ; se pune pe stiva adresa numelui fisierului
    call [fopen]               ; apelarea functiei de deschidere a fisierului
    add esp, 4*2               ; eliberam parametrii de pe stiva; 4 = dimensiunea unui dword; 2 = nr de parametri
```

```
    ; verificam daca functia fopen a creat cu succes fisierul (daca EAX != 0)
    mov [descriptor], eax      ; salvam valoarea returnata de fopen in variabila descriptor_fis
    cmp eax, 0                 ; se compara valoarea din ecx cu 0
    je final                   ; daca e 0 sare la finalul programului
```

```
    ; citim textul in fisierul deschis folosind functia fread
    ; eax = fread(text, 1, len, descriptor_fis)
    push dword [descriptor]
    push dword len
    push dword 1
    push dword sir
    call [fread]
    add esp, 4*4               ; dupa apelul functiei fread EAX contine numarul de caractere citite din fisier
```

```
    mov esi, 0                 ; punem in esi adresa de inceput a sirului
    mov ecx, len                ; se pune in ecx lungimea sirului
```

```
    jecxz final                ; daca ecx = 0 se sare la final
```

```
repet:                               ; inceputul buclei
    mov al, [sir+esi]            ; se pune in al valoarea curenta din sir
    cmp al, ' '                  ; se verifica daca e spatiu sau nu
    je sf                        ; sf = skip flag
    cmp al, 0                    ; conditie de iesire din bucla
    je sf4                       ; sf4 = skip flag 4
    mov bl, al                   ; retinem in bl adresa elementului curent
    sub bl, '0'                  ; convertim din caracter in cifra
    mov al, [curent]             ; in al se pune valoarea numarului curent pentru a-l putea inmulti
    mul byte[ten]                ; se inmulteste cu 10/cream un numar nou
    add al, bl                   ; adunam la rezultat cifra curenta
    mov [curent], al             ; se pune in variabila curent rezultatul obtinut
    jmp sf2                      ; sare la eticheta sf2
```

```

sf:
mov bl, [curent]    ;in bl este pus numarul curent/creat
mov byte[curent], 0 ;se reinitializeaza variabila curent
cmp [minim], bl     ;compara numarul curent/creat cu minimul existent
jb sf2              ;daca nu mai mic decat minimul sare la eticheta sf2
mov [minim], bl     ;altfel actualizam minimul

sf2:
inc esi             ;incrementam registrul esi pentru a trece la urmatorul element/byte din sir
loop repeta         ;repetarea buclei
sf4:                ;eticheta de iesire din bucla
mov bl, [curent]    ;verificarea cazului special in care ultimul numar poate fi mai mic decat minimul
cmp [minim], bl
jb sf3
mov [minim], bl
sf3:

push dword [minim]   ;punem pe stiva valoarea variabilei minim
;push dword [descriptor]
push dword afisare   ;punem pe stiva adresa textului pentru afisare
call [printf]        ;se apeleaza functia de afisare pe ecran
add esp, 4*2         ;eliberam parametrii de pe stiva; 4 = dimensiunea unui dword; 2 = nr de parametri

;push dword [descriptor]
;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

```

The screenshot shows a Windows environment with an assembly editor and a Notepad window. The assembly editor displays the following code:

```

21 descriptor dd 0 ; variabila in care vom salva descriptorul fisierului - necesar pentru a putea face referire la fisier
22 afisare db "Numarul minim din fisier este %d", 0; mesajul care trebuie afisat impreuna cu rezultatul
23 minim dd 255 ; definim o variabila in care retinem minimul
24 curent db 0
25 sir resb 100
26 len equ 100
27 ten db 10
28
29 segment code use32
30 start:
31 push dword 0
32 push dword 0
33 call [fopen]
34 add esp, 4
35
36 ; verificam daca fisierul este deschis
37 mov [descriptor], eax
38 cmp eax, 0
39 je final
40
41 ; citim textul din fisier
42 ; eax = first byte
43 push dword 0
44 push dword 0
45 push dword 0
46 push dword 0
47 push dword 0
48 call [fread]
49 add esp, 4
50
51 mov esi, 0
52 mov ecx, len
53
54 jecxz final
55

```

The Notepad window shows the output of the program, displaying the minimum value found in the file:

```

12 123 13 14 15 16 1 2 3 |

```