Seminar V ASC

Apel de funcții din biblioteci

Pentru a putea apela funcții de bibliotecă (ex din biblioteci .dll sau .lib) trebuie folosită instrucțiunea

call [nume_functie]

Aceasta pune pe stivă adresă următoarei instrucțiuni ce trebuie executată după instrucțiunea *call* (adresa de retur) și face un salt la eticheta *nume_funcție*. Înainte de a apela funcția trebuie transmiși parametrii funcției. Parametrii sunt transmiși funcției cu ajutorul stivei folosind convenția de apel ***CDECL*** (pot fi folosite și alte convenții de apel). Convenția *CDECL* are următoarele caracteristici:

- parametrii sunt transmiși funcției prin stiva de la dreapta la stânga – parametrii sunt puși pe stivă înainte de apel (un element de pe stivă este dublucuvânt);
- funcția întoarce rezultatul în registrul EAX;
- regiștrii EAX, ECX, EDX pot fi modificați de corpul funcției apelate (atenție la valorile stocate în acești regiștrii înainte de apelul funcției);
- eliberarea resurselor (parametrilor de pe stivă) trebuie făcută de codul apelant.

O listă a funcțiilor C run-time (funcții din msvcrt.dll) se poate găsi aici
https://docs.microsoft.com/en-us/cpp/c-runtime-library/reference/crt-alphabetical-function-reference?view=vs-2017

Pentru a afișa informații pe ecran se poate folosi funcția *printf()*. Sintaxa funcției este:

printf (string format, value1, value2, …)

unde *format* este un șir care specifică ce se va afișa pe ecran și *value1*, *value2*... reprezintă valorile afișate (octeți, cuvinte, dublucuvinte, șiruri). Fiecare caracter care apare în *format* va fi afișat pe ecran așa cum este, excepție fac caracterele precedate de simbolul „%”, acestea sunt înlocuite de valorile din lista *value1*, *value2*... Primul caracter din *format* precedat de simbolul % va fi înlocuit de *value1*, al doilea caracter precedat de simbolul % din *format* va fi înlocuit de *value2*, etc. În asamblare orice valoare din lista *value1*, *value2*, ... poate fii o variabilă sau o constantă. Dacă valoarea constantă sau variabilă care trebuie afișată pe ecran nu este un șir, valoarea trebuie pusă pe stivă. Dacă variabila este de tip șir, offset-ul de început al șirului trebuie pus pe stivă. Exemple:

```
printf(“a=%d”, x)              ; va afisa pe ecran “a=[valoarea lui x]”
printf(“%d + %d=%d”,a,b,c)     ; va afisa pe ecran “[valoarea lui a] +
                                ;[valoarea lui b]=[valoarea lui c]”
printf(“%s %d”, s, a)          ; va afisa pe ecran “[sir s] [valoarea lui
                                ; a]”
```

Pentru a citi de la tastatură se poate folosi funcția *scanf()* (atenție la implicațiile de securitate când se folosește *scanf()*). Sintaxa funcției este

scanf (string format, variable1, variable2, …)

unde *format* este un șir care specifică ce se va citi de la tastatură și *value1*, *value2*... reprezintă offset-ul variabilelor (!!!). Șirul *format* ar trebui sa conțină doar caractere precedate de % (ex. %d, %s, etc.). Prima expresie „%” descrie tipul de dată care va fi citită de la tastatură și va fi stocată la offset-ul date de *value1*, a doua expresie „%” descrie tipul de dată care va fi citită de la tastatură și stocată la offset-ul *value2*, etc.. Exemple:

```
scanf(“%d %d”, a, b)   ; citeste doi intregi si ii memorează la
                                ; offset-ul a și b
scanf(“%s”, s)             ; citește un sir si il memoreaza incepand de la
                                ; offset-ul s
```

Ex. 1. Programul de mai jos va afișa pe ecran mesajul „n=” și va citi de la tastatură valoarea numărului n.

```
bits 32
global start
extern exit, printf, scanf        ; exit, printf si scanf sunt functii
                                        ;externe
import exit msvcrt.dll
import printf msvcrt.dll          ; se indica asamblorului unde este functia
                                        ; printf
import scanf msvcrt.dll

segment data use32 class=data
   n dd  0
   message  db "n=", 0    ; un sir in C trebuie terminat cu ZERO
   format  db "%d", 0

segment code use32 class=code
   start:

      ; apel printf(mesaj) => se va afisa pe ecran "n="
      push dword message      ; se pune pe stiva offset-ul sirului
      call [printf]           ; apel printf
      add esp, 4*1      ; eliberare resurse folosite la apel printf
                                        ; 4 = dimensiune dword in octeti
                                        ; 1 = numar parametrii
         ; stiva creste spre adrese mici, un element de pe stiva are
         ; dimensiunea unui dublucuvant

      ; apel scanf(format, n) => se citeste un intreg cu semn
      ; parametrii se pun pe stiva de la dreapta la stanga
      push dword n      ; offset n (NU VALOAREA LUI n)
      push dword format         ; offset format
      call [scanf]          ; apel scanf
```

```
    add esp, 4 * 2      ; eliberare resurse folosite (2 dword)

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

Ex. 2. Să se scrie un program care citește două numere a și b, calculează suma lor și afișează rezultatul pe ecran.

```
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 dd 0
   b dd 0
   result dd 0
   format1 db 'a=', 0   ; format este un sir C
   format2 db 'b=', 0
   readformat db '%d', 0
   printformat db '%d + %d = %d',10, 0

segment code use32 class=code
start:
   ; apel printf("a=")
   push dword format1
   call [printf]
   add esp, 4*1

   ; apel scanf("%d", a)
   push dword a                ; se pune pe stiva offset-ul variabilei!!
   push dword readformat
   call [scanf]
   add esp, 4*2

   ; apel printf("b=")
   push dword format2
   call [printf]
   add esp, 4*1

   ; apel scanf("%d", b)
```

```
push dword b        ; se pune pe stiva offset-ul variabilei!!
push dword readformat
call [scanf]
add esp, 4*2

mov eax, [a]
add eax, [b]
mov [result], eax

; apel printf("%d + %d = %d\n", a, b, result)
push dword [result]    ; pune pe stiva valoarea rezultatului
push dword [b]         ; pune pe stiva valoarea lui b
push dword [a]         ; pune pe stiva valoarea lui a
push dword printformat
call [printf]
add esp,4*4

push dword 0
call [exit]
```

Operatii cu fisiere text

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:

1. Deschiderea fisierului, care poate consta in:
   - Deschiderea unui fisier existent
   - Crearea unui fisier nou
2. Efectuarea operatiilor de scriere si/sau citire
3. Inchiderea fisierului.

Deschiderea unui fisier existent: **fopen**

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

Primul argument al functiei este adresa unui sir de caractere reprezentand numele fisierului. Al doilea argument este adresa unui sir de caractere, reprezentand modalitatea in care se va deschide fisierul.

Mod	Semnificatie	Descriere
r	citire (read)	- Deschide un fisier text pentru citire. Fisierul trebuie sa existe deja pe disc.

w	scriere (write)	- Daca nu exista un fisier cu acel nume, creaza fisierul si il deschide pentru 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.
a	adaugare (append)	- Daca nu exista un fisier cu acel nume, creaza fisierul si il deschide pentru scriere. - 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.
r+	citire si scriere fisier existent	- Deschide un fisier text pentru citire si scriere. Fisierul trebuie sa existe deja pe disc.
w+	citire si scriere	- 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.
a+	citire si adaugare	- 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”)
- Ambele argumente ale functiei *fopen* reprezinta siruri de caractere care trebuie sa se termine cu valoarea 0

Daca fisierul este deschis cu succes, functia fopen va completa in registrul EAX un identificator (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.

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). Pentru a inchide un fisier (deschis in prealabil de functia fopen) se foloseste functia *fclose*.

Scrierea intr-un fisier: **fprintf**.

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

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.

Citirea dintr-un fisier

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

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

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).

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.

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.

Inchiderea unui fisier 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**.

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

Argumentul functiei fclose este descriptorul de fisier (identificatorul) returnal de apelul functiei *fopen*.

Ex. 3. Se citește conținutul unui fișier (a.txt), se adaugă 1 la fiecare octet citit și apoi se scriu octeții rezultați într-un fișier nou (b.txt). **Se redenumeste la finalul scrierii fisierul b.txt in a.txt si se sterge fisierul b.txt din folderul curent**

```
bits 32
global start

; se declara functiile externe necesare pentru a rezolva problema
extern exit, perror, fopen, fclose, fread, fwrite, rename, remove
import exit msvcrt.dll
import fopen msvcrt.dll
import fread msvcrt.dll
import fwrite msvcrt.dll
import fclose msvcrt.dll
import rename msvcrt.dll
import remove msvcrt.dll
import perror msvcrt.dll

segment data use32 class=data
    inputfile db 'a.txt', 0
    outputfile db 'b.txt', 0
    modread db 'r', 0
    modwrite db 'a', 0
    c db 0
    handle1 dd -1
    handle2 dd -1
```

```
    eroare db 'error:', 0

segment code use32 class=code
start:
    ; fopen(string path, string mode) – deschide un fisier aflat la
    ; locatia path in modul specificat. pentru problema mode este “r”
    ; pentru a citi din fisier (“w” pentru a scrie intr-un fisier)
    push dword modread
    push dword inputfile
    call [fopen]
    add esp, 4*2

    ; fopen intoarce in EAX descriptorul de fisier sau 0 (in caz de
    ; eroare)
    ; descriptorul de fisier este un dublucuvant folosit de sistemul
    ; de operare si este cerut de functiile folosite pentru a manipula
    ; fisierul deschis
    mov [handle1], eax ; se salveaza descriptorul intr-o variabila
    cmp eax, 0
    je theend              ; in caz de eroare se incheie executia

    ; fopen(string path, string mode)
    push dword modwrite       ; se deschide fisierul in care se va scrie
                                    ; rezultatul
    push dword outputfile
    call [fopen]
    add esp, 4*2

    ; fopen intoarce in EAX descriptorul de fisier sau 0 (in caz de
    ; eroare)
    mov [handle2], eax
    cmp eax, 0
    je theend

    repeat:
        ;fread(string ptr, integer size, integer n, FILE * handle)
        ; - se citesc de n ori size octeti din fisierul identificat prin
        ; descriptorul pasat ca parametru, octetii cititi sunt salvati in
        ; sirul care incepe la offset-ul ptr
      push dword [handle1]    ; se citeste din descriptor (fisier)
      push dword 1            ; repeta citirea de 1 ori
      push dword 1            ; citeste 1 octet
      push dword c            ; sticheaza octetul in c
      call [fread]
```

```
    add esp, 4*4

    cmp eax, 0          ; functia intoarce 0 in EAX in caz de
                                                ; eroare
    je error
    add byte [c], 1

        ;fwrite(string ptr, integer size, integer n, FILE * handle)
        ; - se scriu de n ori size octeti din sirul ptr in fisierul
        ; identificat prin descriptor
        ; scrie 1 octet in handle2
    push dword [handle2]    ; scrie in fisierul handle2
    push dword 1            ; scrie de 1 ori
    push dword 1            ; scrie 1 octet
    push dword c            ; din sirul c
    call [fwrite]
    add esp, 4*4

    cmp eax, 0
    je error

    jmp repeat


error:
        ; fclose(FILE* handle) – inchide fisierul identificat prin
        ; descriptor
    push dword [handle1]
    call [fclose]
    add esp, 4*1

    push dword [handle2]
    call [fclose]
    add esp, 4*1

        ; remove( string path ) – sterge fisierul de la adresa path
          push dword inputfile
   call [remove]
   add esp, 4*1

        ; rename( string oldname, string newname )
        ; - redenumeste fisierul din oldname in newname (pentru
        ; problemele de la laborator/seminar se presupune ca fisierele
        ; prelucrate se afla in directorul current -> se specifica doar
```

```
        ; numele fisierului nu calea complete catre fisier)

    push dword inputfile
    push dword outputfile
    call [rename]
    add esp, 4*2

    cmp eax, 0   ; intoarce 0 in caz de success, in caz de eroare
                        ; functia intoarce o valoare diferita de 0
    je theend

        ; se afiseaza un mesaj de eroare cu functia "perror()"
        ; apel  perror(eroare)
    push dword eroare
    call [perror]
    add esp, 4*1

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

4. Se citeste de la tastatura un numar n in baza 16 care poate fi reprezentat pe un cuvant (nu se fac validari in acest sens). Sa se deschida fisierul in.txt care contine exact 16 octeti si sa se afiseze pe ecran acei octeti din fisier care se afla pe pozitiile corespunzatoare bitilor 1 din reprezentarea binara a numarului n citit.

Exemplu:
n = F2A1h = 1111 0010 1010 0001b
in.txt = 0123456789abcdef
=> se va afisa pe ecran 0579cdef

Data segment	Code segment
N dd 0 Mod_citire_n db "%x", 0 Mod_afisare db "%c", 0 Nume_fisier db "in.txt", 0 Mod_citire db "r", 0 Descriptor_fisier dd 0 Str times 1 db 0	push dword N push dword Mod_citire_n call [scanf] add esp, 4 * 2 push dword Mod_citire push dword Nume_fisier call [fopen] add esp, 4 * 2 cmp EAX, 0 Jz final

	mov [descriptor_fisier], EAX
	
	Mov ECX, 16
	bucla:
	push ecx
	push dword [descriptor_fisier]
	push dword 1
	push dword 1
	push dword Str
	call [fread]
	add esp, 4 * 4
	
	shr dword [N], 1
	JNC urmatorul
	
	Mov ebx, 0
	Mov bl, [str]
	
	push dword ebx
	push dword mod_afisare
	Call [printf]
	Add esp, 4 * 2
	
	Urmatorul:
	Pop ecx
	Loop bucla
	final:

Tema:

1. Printf(“Suntem in ?? ?? ?? si avem seminar de ??",a,b,c,d)
- data segment?
- code segment?

=> Suntem in 23 Noiembrie 2021 si avem seminar de ASC

2.

```
Ce face urmatoarea secventa?
Este corecta?
Cum trebuie modificata pentru a fi corecta?

data segment
format "%d %s"
a db 1
b db "Ana are mere",0

code segment

push dword a
push dword [b]
push dword format
call [printf]
add esp,4
```

Ce face urmatoarea secventa?

Este corecta?

Cum trebuie modificata pentru a fi corecta?