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\n', 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]
```

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) și apoi se redenumește acest nou fișier cu numele fișierului vechi.

```
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 'w', 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]
```