Apel de funcții din biblioteci

Pentru a putea apela functii din biblioteci se foloseste instructiunea CALL [nume\_functie]

Ce face CALL?

* Pune pe stiva adresa urmatoarei instructiuni ce trebuie executata dupa CALL (adresa de retur)
* Face un salt la eticheta nume\_functie

Conventia de apel CDECL:

* Parametrii care se transmit functiei sunt pusi pe stiva de la dreapta spre stanga
* Functia intoarce rezultatul in registrul EAX
* Registrii EAX, ECX, EDX pot fi modificati de corpul functiei nume\_functie (salvarea pe stiva registrilor cu PUSHAD de exemplu)
* Eliberarea resurselor (parametrilor de pe stiva) se face de catre codul apelant

O lista a functiilor din msvcrt.dll:

* [*https://docs.microsoft.com/en-us/cpp/c-runtime-library/reference/crt-alphabetical-function-reference?view=vs-2017*](https://docs.microsoft.com/en-us/cpp/c-runtime-library/reference/crt-alphabetical-function-reference?view=vs-2017)

|  |  |
| --- | --- |
| printf (format, valoare1, valoare2, ...) - AFISARE PE ECRAN | |
| format = repr. un sir de octeti care specifica ce se va afisa pe ecran  valoare1, valoare2, … = reprezinta valorile afisate (numere, caractere, siruri de caractere ...)  specificatori in format: %c, %s, %d, %u, %x | |
| printf(“a=%d”,a) | **data segment**  format db ‘a=%d’,0  a dd 10  **code segment**  …  push dword [a]  push dword format  call [printf]  add esp, 4\*2 |
| printf(“%d + %d = %d”,a,b,c) | **data segment**  format db ‘%d+%d=%d’,0  a dd 10  b dd 20  c dd 0  **code segment**  …  push dword [c]  push dword [b]  push dword [a]  push dword format  call [printf]  add esp, 4\*4 |
| printf(“Sirul %s are lungimea %d”,s,a) | **data segment**  format db ‘Sirul %s are lungimea %d’,0  s db ‘Arhitectura Sistemelor de Calcul’,0  len equ $-s-1  **code segment**  …  push dword len  push dword s  push dword format  call [printf]  add esp, 4\*3 |
| scanf (format, adresa\_variabila1, adresa\_variabila2, ...) | |
| format = un sir care specifica ce se va citi de la tastatura care trebuie sa contina doar caractere precedate de % (“%d %s” - ASA DA, dar ‘n=%d’ - ASA NU)  adresa\_variabila1, adresa\_variabila2, … = offset-uri ale variabilelor | |
| scanf(“%d %d”,a,b) | **data segment**  format db ‘%d %d’,0  a dd 0  b dd 0  **code segment**  …  push dword b  push dword a  push dword format  call [scanf]  add esp, 4\*3 |
| scanf(“%s”,s) | **data segment**  format db ‘%d %d’,0  s times 11 db 0  **code segment**  …  push dword s  push dword format  call [scanf]  add esp, 4\*2 |

1. Sa se afiseze pe ecran mesajul “n=” si sa se citeasca de la tastatura numarul n.

|  |  |
| --- | --- |
| Data segment | Code segment |
| n dd 0  format db ‘%d’, 0  afisare db ‘n = %d’, 0 | push dword n  push dword format  call [scanf]  add esp, 4 \* 2  push dword [n]  push dword afisare  call [printf]  add esp, 4 \* 2 |

1. Sa se scrie un program care citeste doua numere a si b si calculeaza suma si diferenta lor afisand rezultatele pe ecran

|  |  |
| --- | --- |
| Data segment | Code segment |
| a dd 30  b dd 20  Format1 db “%d %d”,0  Format2 db “Suma este %d,diferenta este %d”,0 | Push dword a  Push dword b  Push dword Format1  Call [scanf]  Add esp ,4\*3  Mov eax,dword [a]  Add eax,dword [b]  Mov ebx,dword [a]  Sub ebx,dword [b]  Push dword ebx  Push dword eax  Push dword Format2  Call [printf]  Add esp,4\*3 |

Operatii cu fisiere text:

Pas 1: Deschiderea fisierului (se deschide un fisier existent sau se creeaza un nou fisier si se descihde)

Pas 2: Prelucram fisierul prin operatii de citire / scriere

Pas 3: Inchidem fisierul deschis la Pas 1

|  |  |
| --- | --- |
| fopen (nume\_fisier, mod\_acces) => returneaza un descriptor de fisiere in EAX > 0 daca deschiderea s-a efectuat cu succes sau =0 in caz contrar | |
| nume\_fisier = adresa unui sir de caractere care reprezinta numele fisierului   * Acesta trebuie sa contina si extensia * Fisierul trebuie sa existe in directorul curent, iar in caz ca nu exista se creaza in directorul curent   mod\_acces = adresa unui sir de caractere care reprezinta modul in care se deschide fisierul   * “r” => fiserul se deschide pentru citire; fisierul exista pe disk * “w” => fisierul se deschide pentru scriere; fisierul exista pe disk sau se creeaza; scrierea se va face de la INCEPUTUL fisierului * “a” => fisierul se deschide pentru scriere; fisierul exista pe disk sau se creeaza; scrierea se va face de la SFARSITUL fisierului * “r+” => citire + scriere fisier existent * “w+” => citire + scriere * “a+” => citire + scriere la final (adaugare) | |
| fopen(nume\_fisier, mod\_acces) | **data segment** nume\_fisier db ‘asc.txt’,0  mod\_acces db ‘w’,0  descriptor dd 0 **code segment**  …  push dword mod\_acces  push dword nume\_fisier call [fopen]  add esp, 4\*2  cmp eax,0  je final  mov [descriptor],eax  ;se prelucreaza fisierul  final:  ;se inchide fisierul |
| fclose(descriptor) | |
| descriptor=fopen(nume\_fisier, mod\_acces)  fclose(descriptor) | **data segment** nume\_fisier db ‘asc.txt’,0  mod\_acces db ‘w’,0  descriptor dd 0 **code segment**  …  push dword mod\_acces  push dword nume\_fisier call [fopen]  add esp, 4\*2  cmp eax,0  je final  mov [descriptor],eax  ;se prelucreaza fisierul  ;se inchide fisierul  push dword [descriptor]  call [fclose]  add esp,4\*1  final: |
| fprintf (descriptor, format, valoare1, valoare2, ….) - SCRIERE IN FISIER | |
| descriptor=fopen(nume\_fisier, mod\_acces)  fprintf (descriptor, format, valoare1, valoare2, ….)  fclose(descriptor)  Format=text | **data segment**  nume\_fisier db ‘asc.txt’,0  mod\_acces db ‘w’,0  descriptor dd 0  Text db “In 27 Noiembrie 2020 se va afisa pe moodle un Sample Quiz”,0  **code segment**    …  push dword mod\_acces  push dword nume\_fisier  call [fopen]  add esp, 4\*2    cmp eax,0  je final    mov [descriptor],eax  ;se prelucreaza fisierul  push dword text  push dword [descriptor]  call [fprintf]  Add esp, 4\*2    ;se inchide fisierul    push dword [descriptor]  call [fclose]  add esp,4\*1  final: |
| fread(str, size, count, descriptor) - CITIREA DIN FISIER | |
| str = adresa unui sir de elemente care se citesc din fisier (in cazul fisierelor text, element=octet => str este un sir de octeti)  size = dimensiunea unui element (in cazul fisierelor text, size=1)  count = reprezinta numarul maxim de elemente care vor fi citite din fisier  => numarul de caractere efectiv citite este returnat in EAX | |
| descriptor=fopen(nume\_fisier,mod\_acces)  fread(str,size,count,descriptor)  fclose(descriptor) | **data segment**  nume\_fisier db ‘asc.txt’,0  mod\_acces db ‘r’,0  descriptor dd 0  len equ 10  text times len db 0  numarCaractere dd 0  **code segment**    …  push dword mod\_acces  push dword nume\_fisier  call [fopen]  add esp, 4\*2    cmp eax,0  je final    mov [descriptor],eax  ;se prelucreaza fisierul  repeta:  push dword [descriptor]  push dword len  push dword 1  push dword text  call [fread]  add esp, 4\*4  cmp eax,0  je gata  add [numarCaractere],eax  jmp repeta  gata:  ;se inchide fisierul    push dword [descriptor]  call [fclose]  add esp,4\*1  final: |

1. Se citeste continutul unui fisier (a.txt), se adauga 1 la fiecare octet citit si se scriu octetii rezultati intr-un fisier nou (b.txt). Apoi, se redenumeste acest fisier cu numele fisierului vechi (in a.txt)

|  |  |
| --- | --- |
| Data segment | Code segment |
| FisierA db ‘a.txt’,0  FisierB db ‘b.txt’,0  ModR db ‘r’,0  ModW db ‘w’,0  DescriptorA dd 0  DescriptorB dd 0  Oct db 0  Format db ‘%c’,0 | Push dword ModR  Push dword FisierA  Call [fopen]  Add esp, 4\*2  Cmp eax, 0  Je Final  Mov [DescriptorA], eax  Push dword ModW  Push dword FisierB  Call [fopen]  Add esp, 4\*2  Cmp eax, 0  Je Final  Mov [DescriptorB], eax  Repeta:  Push dword DescriptorA  Push dword 1  Push dword 1  Push dword Oct  Call [fread]  Add esp, 4\*4  Cmp eax, 0  Je Redenumire  Inc [byte] Oct  Mov eax, 0  Mov al, byte [Oct]  Push eax  Push dword format  Push dword [DescriptorB]  Call [fprintf]  Add esp, 4\*3  Cmp eax, 0  Je Redenumire  Jmp Repeta    Redenumire:  Push dword [DescriptorA]  Call [fclose]  Add esp, 4\*1  Push dword [DescriptorB]  Call [fclose]  Add esp, 4\*1  Push dword FisierA  Call [remove]  Add esp, 4\*1  Push dword FisierA  Push dword FisierB  Call [rename]  Add esp, 4\*2  Final: |
|  |  |

Printf(“Suntem in ?? ?? ?? si avem seminar de ??",a,b,c,d)

* data segment?
* code segment?

=> Suntem in 26 Noiembrie 2020 si avem seminar de ASC