Apel de funcții din biblioteci

* Pentru a apela functii din biblioteci trebuie folosita instructiunea

CALL [nume\_functie]

* CALL – pune pe stiva adresa urmatoarei instructiuni ce trebuie executata dupa revenirea din functie (adresa de retur) si face un salt la eticheta nume\_functie
* Inainte de a apela functie trebuie transmisi parametrii functiei, cu ajutorul stivei, folosind conventia de apel CDECL:
  + Parametrii sunt pusi pe stiva de la dreapta la stanga inainte de apel
  + Functia intoarce rezultatul in registrul EAX
  + Registrii EAX, ECX, EDX pot fi modificati in corpul functiei apelate, prin urmare trebuie sa fim atenti la stocarea valorilor registrilor inainte de apelul functiei (PUSHAD, POPAD)
  + Eliberarea resurselor (a parametrilor de pe stiva) de catre codul apelant
* [*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, ...) - AFISAREA PE ECRAN | |
| format = reprez. Un sir de octeti care specifica ce se va afisa pe ecran  Specificatori in format: %c, %s, %d, %u, %x  valoare1, valoare2, …. = reprez. Valorile afisate (numere, caractere, siruri de caractere) | |
| Printf(“%s are lungimea %d”,text, lungime) | **data segment**  format db ‘%s are lungimea %d’,0  text db ‘Vacanta a fost minunata!’,0  lungime equ $-text-1  **code segment**  …  push dword lungime  push dword text  push dword format  call [printf]  add esp, 4\*3 |
| scanf (format, adresa\_var1, adresa\_var2) - CITIRE DE LA TASTATURA | |
| format = un sir care specifica ce se va citi de la tastatura si trebuie sa contina doar caractere precedate de % (format db “%s %d”,0 - OK; format db “Dati n=%d",0 - NU E OK!!!)  adresa\_var1, adresa\_var2, …. = offset-urile variabilelor in care se salveaza ceea ce se citeste de la tastatura | |
| 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 |
| Operatii cu fisiere text:   1. Deschiderea fisierului (se deschide un fisier existent sau se creeaza un fisier nou) 2. Prelucrarea fisierului (citire / scriere) 3. Inchiderea fisierul deschis | |
| descriptor = fopen (nume\_fisier, mod\_acces) - DESCHIDE FISIERUL si RETURNEAZA in in EAX > 0 daca deschiderea s-a efectuat cu succes si = 0 in caz contrar  nume\_fisier = adresa unui sir de caractere care reprezinta numele fisierului (inclusiv extensia); fisierul trebuie sa existe in directorul curent / se va crea in directorul curent  mod\_acces = adresa unui sir de caractere care reprezinta modul in care se deschide fisierul   * “r” - fisierul se deschide pentru citire; fisierul exista pe disk! * “w” - fisierul se deschide pentru scriere; daca fisierul nu exista se creeaza; scrierea se face de la INCEPUTUL FISIERULUI * “a” - fisierul se deschide pentru scriere; daca fisierul nu exista se creeaza; scrierea se face de la FINALUL FISIERULUI * “r+” - citire + scriere in fisier existent * “w+” - citire + scriere * “a+” - citire + adaugare (scriere la final) | |
| fclose(descriptor) - INCHIDE FISIERUL | |
| 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  push dword [descriptor]  call [fclose]  add esp,4\*1  final: |
| fprintf(descriptor, format, valoare1, valoare2, ...) - SCRIEREA IN FISIER | |
| descriptor=fopen(nume\_fisier, mod\_acces)  fprintf(descriptor, format, val1, val2, ...)  fclose(descriptor) | **Data segment**  nume\_fisier db ‘asc.txt’,0  mod\_acces db ‘w’,0  descriptor dd 0  text db ‘Vacanta a fost minunata!’,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  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, elementul este un octet) => adresa unui sir de octeti  size = dimensiunea unui element (in cazul fisierelor text size=1)  count = numar maxim de elemente care se doresc a fi citite din fisier  descriptor = descriptorul de fisiere returnat de fopen  Se returneaza in registrul EAX numarul efectiv de caractere citite | |
| 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  text times 100 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  bucla:  push dword [descriptor] push dword 100  push dword 1 push dword text call [fread] add esp, 4\*4  cmp eax, 0 je gata  add [numarCaractere],eax  jmp bucla  gata:  push dword [descriptor]  call [fclose]  add esp,4\*1  final: |

Exercitii:

1. Sa se citeasca de la tastatura doua numere si sa se afiseze pe ecran suma si diferenta lor:

|  |  |
| --- | --- |
| Data segment | Code segment |
| a dd 0  b dd 0  suma dd 0  diferenta dd 0  format\_citire db “%d%d”, 0  format\_afisare db “suma: %d, diferenta: %d”, 0 | push dword b  push dword a  push dword format  Call [scanf]  Add esp 4\*3  Mov eax, [a]  Add eax, [b]  Mov [suma], eax  Mov eax, [a]  Sub eax, [b]  Mov [diferenta], eax  Push dword [diferenta]  Push dword [suma]  Push dword format\_afisare  Call [printf]  Add esp, 4\*3 |
|  |  |

1. Se citeste continutul unui fisier (a.txt) si se adauga 1 la fiecare octet citit. Apoi, se scriu octetii rezultati in fisierul b.txt. **Se redenumeste la finalul scrierii fisierul b.txt in a.txt si se sterge fisierul b.txt din folderul curent**

|  |  |
| --- | --- |
| Data segment | Code segment |
| nume\_fisier db “a.txt”, 0  nume\_fisier\_scriere db “b.txt”, 0  mod\_citire db “r”, 0  mod\_adaugare db “a”, 0  descriptor\_fisier resd 1  descriptor\_fisier\_scriere resd 1  format\_afisare db “%c”, 0  str times 1 db 0 | Push dword mod\_citire  Push dword nume\_fisier  Call [fopen]  Add esp, 4 \* 2  Cmp eax, 0  Jz sfarsit  Mov [descriptor\_fisier], eax  Push dword mod\_adaugare  Push dword nume\_fisier\_scriere  Call [fopen]  Add esp, 4 \* 2  Mov [descriptor\_fisier\_scriere], eax  Cmp eax, 0  Jz sfarsit  Bucla:  Push dword [descriptor\_fisier]  Push dword 1  Push dword 1  Push dword str  Call [fread]  Add esp, 4 \* 4  Cmp eax, 0  Je cleanup  Add byte [str], 1  Mov ebx, 0  Mov bl, str  push ebx  Push dword format\_afisare  Push dword [descriptor\_fisier\_scriere]  Call [fprintf]  Add esp, 4 \* 3  Jmp bucla  Cleanup:  Push dword [descriptor\_fisier]  Call [fclose]  Add esp, 4 \* 1  Push dword [descriptor\_fisier\_scriere]  Call [fclose]  Add esp, 4 \* 1  ;remove(path)  Push dword [nume\_fisier]  Call [remove]  Add esp, 4 \* 1  ;rename(oldname, newname)  Push dword [nume\_fisier]  Push dword [nume\_fisier\_scriere]  Call [rename]  Add esp, 4 \* 2  Sfarsit: |

1. 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.

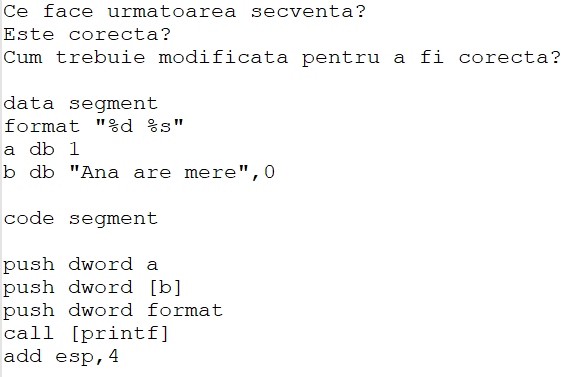
Exemplu:

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 dword [descriptor\_fisier]  push dword 1  push dword 1  push dword Str  call [fread]  add esp, 4 \* 4  shr [N], 1  JNC urmatorul  Mov ebx, 0  Mov bl, [str]  push dword ebx  push dword mod\_afisare  Call [printf]  Add esp, 4 \* 2  Urmatorul:  Loop bucla  final: |



Ce face urmatoarea secventa?

Este corecta?

Cum trebuie modificata pentru a fi corecta?

data segment

format **DB** "%d %s" **,0**

a dd 1

b db "Ana are mere",0

code segment

~~push dword a~~ push dword b

~~push dword [b]~~ push dword [a]

push dword format

call [printf]

add esp,4**\*3**