Seminar 5 – Apel de functii din biblioteci

; nume\_functie(a,b,c,d,e, ...)

Call [nume\_functie]

CALL = se pune pe stiva adresa urmatoarei instructiuni ce trebuie executata dupa instructiunea CALL (adresa de revenire) + se face un salt la eticheta nume\_functie

Pasi:

* Inainte de apel se face transmiterea parametrilor prin intermediul stivei (conventia de apel CDECL), a.i. parametrii sunt transmisi in functie prin intermediul stivei de la dreapta spre stanga
* Functia intoarce rezultatul in registrul EAX
* Registrii EAX, ECX, EDX pot fi modificati in functie de functia apelata => salvarea valorile din registrii inainte de apel
* Eliberarea resurselor (a parametrilor) la revenirea din call

FUNCTII

<https://docs.microsoft.com/en-us/cpp/c-runtime-library/reference/crt-alphabetical-function-reference?view=msvc-170&viewFallbackFrom=vs-2017>

|  |  |
| --- | --- |
| AFISAREA PE ECRAN | |
| Printf(format, val1, val2, ...) | Printf(“a=%d”,16)  Printf(“Suma este %d+%d=%d”,a,b,c)  Printf(“Azi e %d %s %d”,26,”noiembrie”,2021) |
| CITIREA DE LA TASTATURA | |
| Scanf(format,var1,var2) | Scanf(“%d”,n)  Scanf(“%s %d”,s,l) |

1. Sa se scrie un program in limbaj de asamblare care va afisa pe ecran mesajul “Data de azi este: ” si va citi de la tastatura un sir de maxim 30 de caractere de tipul “26 Noiembrie 2021”

|  |
| --- |
| **bits 32**; assembling for the 32 bits architecture    ; declare the EntryPoint (a label defining the very first instruction of the program)  **global start**    ; declare external functions needed by our program  **extern exit, printf, scanf**; tell nasm that exit exists even if we won't be defining it  **import exit msvcrt.dll**; exit is a function that ends the calling process. It is defined in msvcrt.dll  ; msvcrt.dll contains exit, printf and all the other important C-runtime specific functions  **import printf msvcrt.dll**  **import scanf msvcrt.dll**    ; our data is declared here (the variables needed by our program)  **segment data use32 class=data**  **Message db “Data de azi este: “, 0**  **Format db “%s”, 0**  **Read\_message times 31 db 0**    ; our code starts here  **segment code use32 class=code**  **start:**  **Push dword Message**  **Push dword Format**  **Call [printf]**  **Add esp, 4\*2**  **Push dword Read\_message**  **Push dword Format**  **Call [scanf]**  **Add esp, 4\*2**    ; exit (0)  **push dword 0**; push the parameter for exit onto the stack  **Call [exit]**; call exit to terminate the program |

1. Sa se scrie un program in limbaj de asamblare care citeste de la tastatura doua numere a si b in baza 16 reprezentare pe dublucuvant, calculeaza suma tuturor octetilor lor si afiseaza rezultatul pe ecran

|  |
| --- |
| **bits 32**; assembling for the 32 bits architecture    ; declare the EntryPoint (a label defining the very first instruction of the program)  **global start**    ; declare external functions needed by our program  **extern exit, printf, scanf**; tell nasm that exit exists even if we won't be defining it  **import exit msvcrt.dll**; exit is a function that ends the calling process. It is defined in msvcrt.dll  ; msvcrt.dll contains exit, printf and all the other important C-runtime specific functions  **import printf msvcrt.dll**  **import scanf msvcrt.dll**    ; our data is declared here (the variables needed by our program)  **segment data use32 class=data**  **A resd 1**  **B resd 1**  **s1 db 0**  **format db “%x”,0**  ; our code starts here  **segment code use32 class=code**  **start:**  **Push dword A**  **Push dword format**  **Call [scanf]**  **Add esp,4\*2**  **Push dword B**  **Push dword format**  **Call[scanf]**  **Add esp,4\*2**    **Mov Eax,[A]**  **Mov Ebx,[B]**  **Mov Ecx,4**  **Repeta:**  **Add [s1],Al**  **Add [s1],Bl**  **Push ecx**  **Mov cl,8**  **Shr Eax,cl**  **Shr Ebx,cl**  **Pop ecx**  **Loop Repeta**  **Mov eax,0**  **Mov al, [s1]**  **Push dword eax**  **Push dword format**  **Call [printf]**  **Add esp,4\*2**  ; exit (0)  **push dword 0**; push the parameter for exit onto the stack  **Call [exit]**; call exit to terminate the program |

Operatii cu fisiere text

1. Deschidere fisier => descriptor de fisiere (fd)
2. Efectuare operatiile de citire / scriere
3. Inchidere fisier deschis

|  |  |
| --- | --- |
| DESCHIDERE FISIER | |
| fd=fopen(nume\_fisier,mod\_acces) | Nume\_fisier = numele si extensia fisier din directorul curent  Mod\_acces =  “r” - citire (!!! Fis trebuie sa existe)  “w” - scriere  1) daca fis exista, se deschide pt scriere se sterge continutul lui si se scrie de la inceput  2) daca fis nu exista, se creeaza si se deschide pt scriere  “a” - scriere  1) daca fis exista, se deschide pt scriere si se scrie de la sfarsit  2) daca fis nu exista, se creeaza si se deschide pt scriere  “r+”  “w+”  “a+”  **=> functia returneaza in EAX un descriptor de fisier nenul sau 0 in caz de eroare** |
| SCRIERE IN FISIER | |
| fprintf(fd,format,val1,val2) | Fd = descriptorul de fisier returnat de fopen  Restul argumentelor sunt ca la printf |
| CITIRE DIN FISIER | |
| fread(str,size,count,fd) | Str = adresa unui buffer care se completeaza cu datele citite  Size = 1  Count = numarul maxim de elemente citite  Fd= descriptorul de fisier returnat de fopen  **=> in eax se returneaza numarul efectiv de caractere citite** |
| INCHIDERE FISIER | |
| fclose(fd) | Fd= descriptorul de fisier returnat de fopen |

3. Se citeste continutul unui fisier (a.txt), se adauga 1 la fiecare octet si apoi se scriu octetii in fisierul b.txt. Se redenumeste la final b.txt in a.txt si se sterge b.txt din folderul curent

|  |
| --- |
| **bits 32**; assembling for the 32 bits architecture    ; declare the EntryPoint (a label defining the very first instruction of the program)  **global start**    ; declare external functions needed by our program  **extern exit, printf, scanf, fopen, fclose, fprintf, fread, rename, remove**; tell nasm that exit exists even if we won't be defining it  **import exit msvcrt.dll**; exit is a function that ends the calling process. It is defined in msvcrt.dll  ; msvcrt.dll contains exit, printf and all the other important C-runtime specific functions  **import printf msvcrt.dll**  **import scanf msvcrt.dll**  **import fopen msvcrt.dll**  **import fclose msvcrt.dll**  **import fprintf msvcrt.dll**  **import fread msvcrt.dll**  **import rename msvcrt.dll**  **import remove msvcrt.dll**  ; our data is declared here (the variables needed by our program)  **segment data use32 class=data**  ; our code starts here  **A db 0**  **Format db “%c”, 0**  **Nume\_fisier db “a.txt”, 0**  **Nume\_fisier\_w db “b.txt”, 0**  **Mod\_acces db “r”, 0**  **Mod\_acces\_w db “w”, 0**  **Descriptor dd 0**  **Descriptor\_w dd 0**  **segment code use32 class=code**  **start:**  **Push dword Mod\_acces**  **Push dword Nume\_fisier**  **Call [fopen]**  **Add ESP, 4 \* 2**    **Mov [Descriptor], EAX**  **Cmp EAX, 0**  **Jz Final**  **Push dword Mod\_acces\_w**  **Push dword Nume\_fisier\_w**  **Call [fopen]**  **Add ESP, 4 \* 2**    **Mov [Descriptor\_w], EAX**    **Cmp EAX, 0**  **Jz Final**    **Repeta:**    **Push dword [Descriptor]**  **Push dword 1**  **Push dword 1**  **Push dword A**  **Call [fread]**  **Add ESP, 4 \*4**    **Cmp EAX, 0**  **Je Final\_parcurgere**    **Add byte[A], 1**    **Mov EAX, 0**  **Mov Al, [A]**    **Push dword EAX**  **Push dword Format**  **Push dword [Descriptor\_w]**  **Call [fprintf]**  **Add ESP, 4 \* 3**  **Jmp repeta**    **Final\_parcurgere:**  **Push dword [Descriptor]**  **Call [fclose]**  **Add ESP, 4**  **Push dword [Descriptor\_w]**  **Call [fclose]**  **Add ESP, 4**  **;remove(path)**  **Push dword Nume\_fisier**  **Call [remove]**  **Add ESP, 4**  **;rename(oldname,newname)**  **Push dword Nume\_fisier**  **Push dword Nume\_fisier\_w**  **Call [rename]**  **Add ESP, 4 \* 2**  **Final:**  ; exit (0)  **push dword 0**; push the parameter for exit onto the stack  **Call [exit]**; call exit to terminate the program |

1. Se citeste de la tastatura un numar N in baza 16 care poate fi reprezentat pe un cuvant. Sa se deschida fisierul in.txt care contine exact 16 octeti. Sa se afiseze pe ecran acei octeti din fisier care se afla pe pozitiile corespunzatoare bitilor 1 din reprezentarea binara a numarului N citit

Ex: N=F2A1h=1111 0010 1010 0001b  
in.txt=**0**1234**5**6**7**8**9**ab**cdef**

=> se va afisa 0579cdef

|  |
| --- |
| **bits 32**; assembling for the 32 bits architecture    ; declare the EntryPoint (a label defining the very first instruction of the program)  **global start**    ; declare external functions needed by our program  **extern exit, printf, scanf, fopen, fclose, fprintf, fread, rename, remove**; tell nasm that exit exists even if we won't be defining it  **import exit msvcrt.dll**; exit is a function that ends the calling process. It is defined in msvcrt.dll  ; msvcrt.dll contains exit, printf and all the other important C-runtime specific functions  **import printf msvcrt.dll**  **import scanf msvcrt.dll**  **import fopen msvcrt.dll**  **import fclose msvcrt.dll**  **import fprintf msvcrt.dll**  **import fread msvcrt.dll**  **import rename msvcrt.dll**  **import remove msvcrt.dll**  ; our data is declared here (the variables needed by our program)  **segment data use32 class=data**  ; our code starts here  **N resd 1**  **Format “%x”, 0**  **NumeFisier “in.txt”, 0**  **ModAcces db “r”, 0**  **Descriptor dd 0**  **Buffer db 0, 0**    **segment code use32 class=code**  **start:**  Push dword n  Push dword Format  Call [scanf]  Add esp, 4 \* 2    Push dowrd ModAcces  Push dword NumeFisier  Call [fopen]  Add esp, 4 \* 2  Cmp eax, 0  Je Final  Mov [Descriptor],eax  Mov ecx, 16  Citire:  Push ecx  Push dword [Descriptor]  Push dword 1  Push dword 1  Push dword Buffer  Call [fread]  Add esp,4\*4  ;[buffer] - caracterul citit  ;ne uitam pe pozitia coresp in N a.i sa shiftam spre dreapta cu o pozitie pentru a identifica daca avem 1 sau 0 in bitul curent  ;TEMA afisarea conditionata  Pop ecx  Loop Citire    **Final:**  ; exit (0)  **push dword 0**; push the parameter for exit onto the stack  **Call [exit]**; call exit to terminate the program |