**Lab 8 – Apeluri de functii (citire de la tastatura si afisare pe ecran)**

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.***

**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 – se recomanda salvarea registrilor inainte de folosirea functiilor externe);

**pushad – pune pe stiva toti reg generali**

**popad – scoate de pe stiva toti reg generali**

* eliberarea resurselor (parametrilor de pe stivă) trebuie făcută de codul apelant (adica noi eliberam si parametrii de pe stiva).

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.

**%d zecimal**

**%x hexazecimal**

**%c caracter**

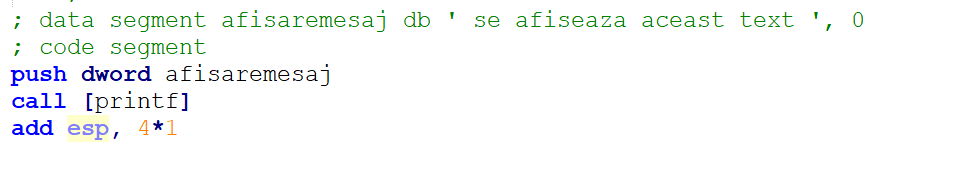
**%s string**

În asamblare orice valoare din lista *value1*, *value2*, ... poate fi 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.

Ex:



Pentru a citi de la tastatură se poate folosi funcția *scanf()*.

Sintaxa funcției este

**scanf (string format, variable1, variable2, … )**

unde *format* este un șir care specifică ce se va citi de la tastatură și *vaariable1*, *vaariable2*... 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 *variable1*, a doua expresie „%” descrie tipul de dată care va fi citită de la tastatură și stocată la offset-ul *variable 2*, etc..

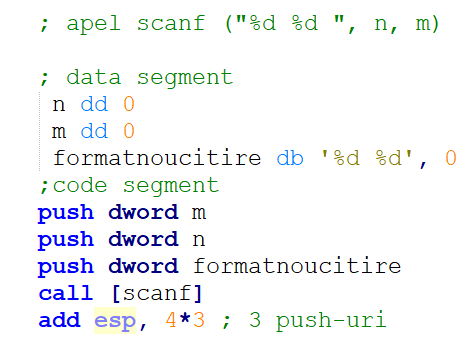
Exemple:

scanf(“%d %d %d”, a, b, c) *; citeste trei intregi si ii memorează la*

*; offset-ul a și b si c*

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 ; ***un sir in C trebuie terminat cu ZERO***

**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’, 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]

Atentie, pentru a rula, nu este necesar sa intrati in debugger: Puteti alege direct RUN Program.

