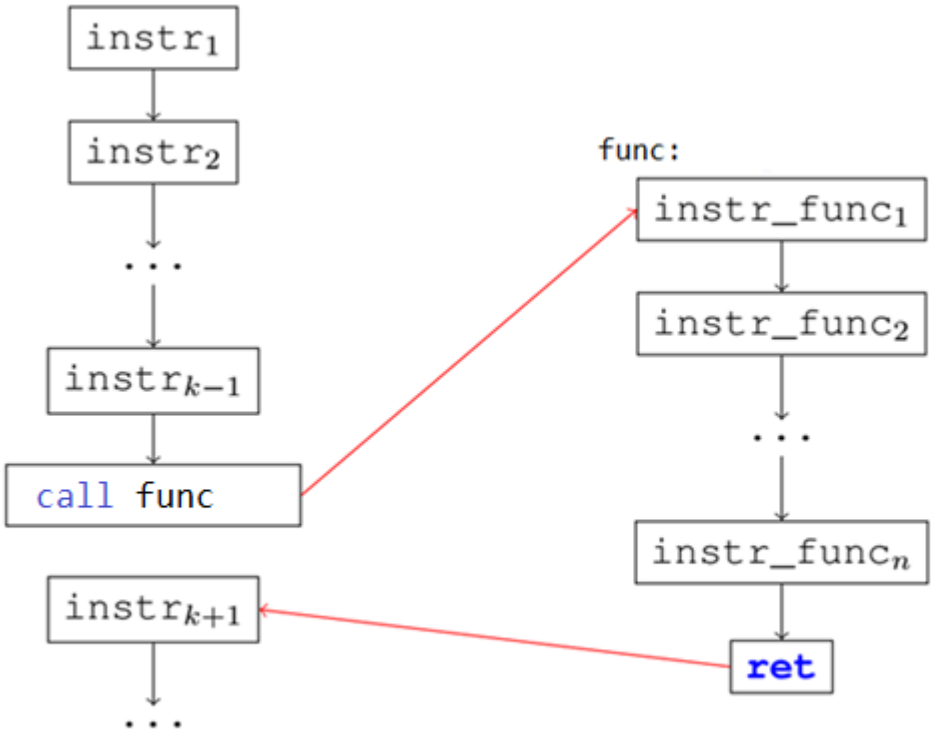
Laborator 5 - Suport teoretic

Apeluri de functii

Biblioteci de functii (function libraries)

Chiar daca limbajul de asamblare foloseste in mod direct componentele hardware ale sistemului, exista portiuni de cod utilizate frecvent, care ar fi impractic sa fie scrise de catre programator de fiecare data. De exemplu, comunicarea cu dispozitivele de intrare/iesire, care presupune de cele mai multe ori protocoale complexe. Unul dintre rolurile sistemului de operare este acela de a abstractiza masina hardware pentru programator, punand la dispozitie multiple biblioteci de functii ce pot fi apelate pentru anumite operatii frecvente, cum ar fi afisarea datelor intr-un anumit format, gasirea unui subsir intr-un sir sau diverse functii matematice.

Utilizarea unei functii presupune saltul la portiunea de cod corespunzatoare functiei, executia acestui cod, urmata de revenirea la instructiunea de dupa cea care a apelat functia, ca in figura de mai jos:



Utilizarea functiilor externe. Conventii de apel

Pentru apelul unei functii se foloseste instructiunea call.

Sintaxa:

```
call <nume_functie_sistem>
```

Semantica:

Aceasta instructiune pune adresa instructiunii urmatoare pe stiva, apoi sare la inceputul functiei apelate. Punerea pe stiva a adresei instructiunii urmatoare (adresa de revenire) se face pentru ca la finalul executiei sa se poata reveni la codul apelant.

Apelarea unei functii sistem. Parametrii

De cele mai multe ori, functiile pot primi parametri, si pot intoarce rezultate. Exista mai multe conventii pentru a face aceste lucruri, dintre care in acest laborator vom folosi conventia **cdecl**.

O conventie de apel nu tine de sintaxa limbajului de asamblare, ci este un ”contract” intre autorul functiei si utilizatorii acesteia, ce specifica modul de transmitere a parametrilor, respectiv de intoarcere a rezultatului.

Conventia cdecl

- Argumentele se pun pe stiva de la dreapta la stanga
- Rezultatul implicit este returnat de catre functia sistem in EAX
- Registrii EAX, ECX, EDX pot fi folositi in interiorul functiei (isi pot modifica valoarea!)
- **Atentie deci, daca doriti pastrarea valorilor initiale din EAX, ECX, EDX sa salvati aceste valori (in variabile de memorie, pe stiva sau in alti registrii) inainte de apelul functiei**
- Functia nu va elibera argumentele de pe stiva. Este responsabilitatea programatorului sa faca acest lucru dupa apelul functiei
- Exemple: printf, scanf

Functii standard din msvcrt

Afisarea pe ecran

Pentru afisarea unui text pe ecran, ce respecta un anumit format, se foloseste functia *printf*.

Sintaxa functiei *printf* in limbaj de programare de nivel inalt este:

```
int printf(const char * format, variabila_1, constanta_2, ...);
```

Functia *printf* respecta conventia *cdecl*.

Primul argument al functiei este un sir de caractere ce contine formatul afisarii, urmat de un numar de argumente (valori constante sau nume de variabile) egal cu cel specificat in cadrul formatului.

Sirul de caractere transmis in parametrul format poate contine anumite marcaje de formatare, ce incep cu caracterul '%', care vor fi inlocuite de valorile specificate in urmatoarele argumente, formatate corespunzator.

Specificator	Ce se afiseaza	Exemplu	Dimensiune reprezentare valoare
c	Caracter	a	byte
d sau i	Intreg zecimal cu semn	392	dword
u	Intreg zecimal fara semn	7235	dword
x	Numar in hexazecimal fara semn	7fa	dword
s	String (sir de caractere, terminat cu 0)	exemplu	sir de bytes terminat in 0

Exemple:

Afisarea unui mesaj

- In limbaj de programare de nivel inalt:

```
printf("Ana are mere");
```

- Echivalentul in limbaj de asamblare:

```
segment data use32 class=data
        mesaj  db "Ana are mere", 0  ; definim mesajul
segment code use32 class=code
        ; ...
         push dword mesaj  ; punem parametrul pe stiva
         call [printf]       ; apelam functia printf
```

```
          add esp, 4 * 1     ; eliberam parametrii de pe stiva
         ; ...
```

Afisarea unui numar natural cu semn in baza 10

- In limbaj de programare de nivel inalt:

```
printf("%d", -17);
```

- Echivalentul in limbaj de asamblare:

```
segment data use32 class=data
        format  db "%d", 0  ; definim formatul
segment code use32 class=code
        ; ...
         push dword -17  ; punem parametrii pe stiva de la dreapta la
stanga
         push dword format
         call [printf]       ; apelam functia printf
         add esp, 4 * 2     ; eliberam parametrii de pe stiva
        ; ...
```

Afisarea unui numar natural in baza 16

- In limbaj de programare de nivel inalt:

```
printf("%x", 0xAB);
```

- Echivalentul in limbaj de asamblare:

```
segment data use32 class=data
        format  db "%x", 0  ; definim formatul
segment code use32 class=code
        ; ...
         push dword 0xAB  ; punem parametrii pe stiva de la dreapta la
stanga
         push dword format
         call [printf]       ; apelam functia printf
         add esp, 4 * 2     ; eliberam parametrii de pe stiva
        ; ...
```

Afisarea unui mesaj care contine un numar natural in baza 10, stocat intr-o variabila n

- In limbaj de programare de nivel inalt:

```
printf("Ana are %d mere", n);
```

- Echivalentul in limbaj de asamblare:

```
segment data use32 class=data
        n dd 7
        format  db "Ana are %d mere", 0  ; definim formatul
segment code use32 class=code
        ; ...
         push dword [n]  ; punem pe stiva valoarea lui n
         push dword format
         call [printf]       ; apelam functia printf
         add esp, 4 * 2     ; eliberam parametrii de pe stiva
        ; ...
```

Afisarea unui mesaj care contine mai multe numere in baza 10.

- In limbaj de programare de nivel inalt:

```
printf("Ana are %d mere si %d pere", 7, 8);
```

- Echivalentul in limbaj de asamblare:

```
segment data use32 class=data
        format  db "Ana are %d mere si %d pere", 0  ; definim formatul
segment code use32 class=code
        ; ...
         push dword 8  ; punem parametrii pe stiva
         push dword 7
         push dword format
         call [printf]       ; apelam functia printf
         add esp, 4 * 3     ; eliberam parametrii de pe stiva
        ; ...
```

Citirea de la tastatura

Pentru a citi date de la tastatura se foloseste functia *scanf*.

Sintaxa functiei *scanf* in limbaj de programare de nivel inalt este:

```
int scanf(const char * format, adresa_variabila_1, ...);
```

Sintaxa acestei functii este similara cu cea a functiei printf. Diferenta majora consta in faptul ca argumentele sale nu trebuie sa fie valori constante sau continuturi de variabile, ci numai adrese in memorie, unde se vor stoca valorile citite.

Exemplu:

Citirea unui numar natural in variabila n

- In limbaj de programare de nivel inalt:

```
scanf("%d", &n);

  &n reprezinta adresa variabilei n in care functia scanf va completa valoarea citita de la tastatura
```

- Echivalentul in limbaj de asamblare:

```
segment data use32 class=data
        n dd  0         ; definim variabila n
        format  db "%d", 0  ; definim formatul
segment code use32 class=code
        ; ...
        push dword n        ; punem parametrii pe stiva de la dreapta la stanga
        push dword format
        call [scanf]        ; apelam functia scanf pentru citire
        add esp, 4 * 2      ; eliberam parametrii de pe stiva
        ; ...
```

Laborator 5 - Exemple

Apeluri de functii sistem

Afisarea unui mesaj

```
; Codul de mai jos va afisa mesajul „Ana are 17 mere”
bits 32
global start

; declararea functiilor externe folosite de program
extern exit, printf          ; adaugam printf ca functie externa
import exit msvcrt.dll
import printf msvcrt.dll     ; indicam asamblorului ca functia printf se
gaseste in libraria msvcrt.dll

segment data use32 class=data
; sirurile de caractere sunt de tip byte
format db "Ana are %d mere", 0  ; %d va fi inlocuit cu un numar
                                ; sirurile de caractere pt functiile C trebuie
terminate cu valoarea 0
segment  code use32 class=code
    start:
        mov eax, 17

        ; vom apela printf(format, 17) => se va afisa: „Ana are 17 mere”
        ; punem parametrii pe stiva de la dreapta la stanga
        push dword eax
        push dword format ; ! pe stiva se pune adresa string-ului, nu valoarea
        call [printf]       ; apelam functia printf pentru afisare
        add esp, 4 * 2      ; eliberam parametrii de pe stiva; 4 = dimensiunea
unui dword; 2 = nr de parametri

        ; exit(0)
        push dword 0       ; punem pe stiva parametrul pentru exit
        call [exit]        ; apelam exit pentru a incheia programul
```

Citirea unui numar de la tastatura

```
; Codul de mai jos va afisa mesajul ”n=”, apoi va citi de la tastatura
valoarea numarului n.
bits 32

global start

; declararea functiilor externe folosite de program
extern exit, printf, scanf ; adaugam printf si scanf ca functii externa
import exit msvcrt.dll
import printf msvcrt.dll     ; indicam asamblorului ca functia printf se
gaseste in libraria msvcrt.dll
```

```
import scanf msvcrt.dll     ; similar pentru scanf

segment data use32 class=data
        n dd  0         ; in aceasta variabila se va stoca valoarea citita de la
tastatura
    ; sirurile de caractere sunt de tip byte
        message  db "n=", 0  ; sirurile de caractere pentru functiile C
trebuie sa se termine cu valoarea 0 (nu caracterul)
        format  db "%d", 0  ; %d <=> un numar decimal (baza 10)

segment code use32 class=code
    start:

        ; vom apela printf(message) => se va afisa "n="
        ; punem parametrii pe stiva
        push dword message ; ! pe stiva se pune adresa string-ului, nu
valoarea
        call [printf]       ; apelam functia printf pentru afisare
        add esp, 4*1        ; eliberam parametrii de pe stiva ; 4 = dimensiunea
unui dword; 1 = nr de parametri

        ; vom apela scanf(format, n) => se va citi un numar in variabila n
        ; punem parametrii pe stiva de la dreapta la stanga
        push dword n        ; ! adresa lui n, nu valoarea
         push dword format
        call [scanf]        ; apelam functia scanf pentru citire
        add esp, 4 * 2      ; eliberam parametrii de pe stiva
                            ; 4 = dimensiunea unui dword; 2 = nr de parametri

        ; exit(0)
        push dword 0       ; punem pe stiva parametrul pentru exit
        call [exit]        ; apelam exit pentru a incheia programul
```

Salvarea valorilor din registrii inaintea apelurilor de functii sistem

```
; Codul de mai jos va calcula rezultatul unor operatii aritmetice in registrul
EAX, va salva valoarea registrilor, apoi va afisa valoarea rezultatului si va
restaura valoarea registrilor.
bits 32

global start

; declararea functiilor externe folosite de program
extern exit, printf  ; adaugam printf ca functie externa
import exit msvcrt.dll
import printf msvcrt.dll    ; indicam asamblorului ca functia printf se
gaseste in libraria msvcrt.dll

segment data use32 class=data
        ; sirurile de caractere sunt de tip byte
        format  db "%d", 0  ; %d <=> un numar decimal (baza 10)

segment code use32 class=code
        start:
                ; vom calcula 20 + 123 + 7 in EAX
```

```
                mov eax, 20
                add eax, 123
                add eax, 7          ; eax = 150 (baza 10) sau 0x96 (baza 16)

                ; salvam valoarea registrilor deoarece apelul functiei sistem
printf va modifica valoarea acestora
                ; folosim instructiunea PUSHAD care salveaza pe stiva valorile
mai multor registrii, printre care EAX, ECX, EDX si EBX
                ; in acest exemplu este important sa salvam doar valoarea
registrului EAX, dar instructiunea poate fi aplicata generic
                PUSHAD

                ; vom apela printf(format, eax) => vom afisa valoarea din eax
                ; punem parametrii pe stiva de la dreapta la stanga
                push dword eax
                push dword format ; ! pe stiva se pune adresa string-ului, nu
valoarea
                call [printf]       ; apelam functia printf pentru afisare
                add esp, 4*2        ; eliberam parametrii de pe stiva ; 4 =
dimensiunea unui dword; 2 = nr de parametri

                ; dupa apelul functiei printf registrul EAX are o valoare
setata de aceasta functie (nu valoarea 150 pe care am calculat-o la inceputul
programului)
                ; restauram valoarea registrilor salvati pe stiva la apelul
instructii PUSHAD folosind instructiunea POPAD
                ; aceasta instructiune ia valori de pe stiva si le completeaza
in mai multi registrii printre care EAX, ECX, EDX si EBX
                ; este important ca inaintea unui apel al instructiunii POPAD
sa ne asiguram ca exista suficiente valori
                ; pe stiva pentru a fi incarcate in registrii (de exemplu ca
anterior a fost apelata instructiunea PUSHA)
                POPAD

                ; acum valoarea registrului EAX a fost restaurata la valoarea
de dinaintea apelului instructiunii PUSHAD (in acest caz valoarea 150)

                ; exit(0)
                push dword 0       ; punem pe stiva parametrul pentru exit
                call [exit]        ; apelam exit pentru a incheia programul
```

Laborator 5 - Probleme propuse

Probleme propuse

1. Sa se citeasca de la tastatura doua numere (in baza 10) si sa se calculeze produsul lor. Rezultatul inmultirii se va salva in memorie in variabila "rezultat" (definita in segmentul de date).
2. Sa se citeasca de la tastatura doua numere a si b (in baza 10) si sa se calculeze a/b. Catul impartirii se va salva in memorie in variabila "rezultat" (definita in segmentul de date). Valorile se considera cu semn.
3. Se dau doua numere naturale a si b (a, b: dword, definite in segmentul de date). Sa se calculeze suma lor si sa se afiseze in urmatorul format:
   "<a> + <b> = <result>"
   Exemplu: "1 + 2 = 3"
   Valorile vor fi afisate in format decimal (baza 10) cu semn.
4. Se dau doua numere naturale a si b (a, b: word, definite in segmentul de date). Sa se calculeze produsul lor si sa se afiseze in urmatorul format:
   "<a> * <b> = <result>"
   Exemplu: "2 * 4 = 8"
   Valorile vor fi afisate in format decimal (baza 10) cu semn.
5. Se dau doua numere naturale a si b (a, b: word, definite in segmentul de date). Sa se calculeze a/b si sa se afiseze catul si restul impartirii in urmatorul format:
   "Cat = <cat>, rest = <rest>"
   Exemplu: pentru a=23 si b=10 se va afisa: "Cat = 2, rest = 3"
   Valorile vor fi afisate in format decimal (baza 10) cu semn.
6. Se dau doua numere naturale a si b (a: dword, b: word, definite in segmentul de date). Sa se calculeze a/b si sa se afiseze catul impartirii in urmatorul format:
   "<a>/<b> = <cat>"
   Exemplu: pentru a = 200 si b = 5 se va afisa: "200/5 = 40"
   Valorile vor fi afisate in format decimal (baza 10) cu semn.
7. Se dau doua numere natural a si b (a: dword, b: word, definite in segmentul de date). Sa se calculeze a/b si sa se afiseze restul impartirii in urmatorul format:
   "<a>/<b> = <cat>"
   Exemplu: pentru a = 23 si b = 5 se va afisa: "23 mod 5 = 3"
   Valorile vor fi afisate in format decimal (baza 10) cu semn.
8. Se da un numar natural a (a: dword, definit in segmentul de date). Sa se citeasca un numar natural b si sa se calculeze: a + a\b. Sa se afiseze rezultatul operatiei. Valorile vor fi afisate in format decimal (baza 10) cu semn.
9. Sa se citeasca de la tastatura doua numere a si b (in baza 10) si sa se calculeze: (a+b) / (a-b). Catul impartirii se va salva in memorie in variabila "rezultat" (definita in segmentul de date). Valorile se considera cu semn.
10. Sa se citeasca de la tastatura un numar in baza 10 si sa se afiseze valoarea acelui numar in baza 16.
    Exemplu: Se citeste: 28; se afiseaza: 1C
11. Sa se citeasca de la tastatura un numar in baza 16 si sa se afiseze valoarea acelui numar in baza 10.
    Exemplu: Se citeste: 1D; se afiseaza: 29
12. Se da un numar natural negativ a (a: dword). Sa se afiseze valoarea lui in baza 10 si in baza 16, in urmatorul format: "a = <base_10> (baza 10), a = <base_16> (baza 16)"

13. Sa se citeasca de la tastatura doua numere a si b (in baza 10) si sa se calculeze: (a+b) * (a-b). Rezultatul inmultirii se va salva in memorie in variabila "rezultat" (definita in segmentul de date).
14. Sa se citeasca de la tastatura doua numere a si b (in baza 16) si sa se calculeze: a+b. Sa se afiseze rezultatul adunarii in baza 10.
15. Sa se citeasca de la tastatura doua numere a si b (in baza 10) si sa se calculeze: a+b. Sa se afiseze rezultatul adunarii in baza 16.
