ARHITECTURA SISTEMELOR DE CALCUL
– Seminar 5 –

1. Apeluri de funcții
2. Operații cu fișiere de tip text

1. APELURI DE FUNCȚII

O *bibliotecă de funcții (a function library)* este un fișier care conține definițiile (în cod mașină) unui grup de funcții. Un exemplu de bibliotecă de funcții este *msvcrt.dll (Microsoft Visual C Runtime).*

Lista completă a funcțiilor definite în biblioteca de funcții *msvcrt.dll* poate fi consultată la această adresă:

https://learn.microsoft.com/en-us/cpp/c-runtime-library/reference/crt-alphabetical-function-reference

1.a. Instrucțiunea `CALL`

Apelul unei funcții de bibliotecă definite într-o bibliotecă de funcții externă se realizează cu ajutorul instrucțiunii **`CALL`** astfel:

```
call [nume_funcție]
```

Instrucțiunea **`CALL`** execută un *„salt cu revenire”*, prin urmare, execuția sa va produce următoarele efecte:

– se pune în vârful stivei offset-ul instrucțiunii imediat următoare (valoarea curentă din registrul EIP);

– se execută un salt de tip FAR în segmentul de cod care conține definiția funcției apelate.

Observații

Înainte de a efectua apelul unei funcții definite într-o bibliotecă de funcții externă este OBLIGATORIU să indicăm asamblorului unde se găsește definiția funcției apelate. În cazul apelului în NASM a unor funcții definite în libraria *msvcrt.dll (Microsoft Visual C Runtime)* acest lucru se realizează cu ajutorul directivelor `extern` și `import`, astfel:

– indicăm asamblorului că aceste funcții (simboluri) sunt definite într-o librărie externă (nu sunt definite în segmentul de cod curent):

```
extern nume_functie1, nume_functie2, nume_functie3
```

– pentru FIECARE dintre funcțiile enumerate mai sus, indicăm asamblorului de unde va *„importa”* acele funcții (care este numele librăriei în care sunt definite):

```
import nume_functie1 msvcrt.dll
import nume_functie2 msvcrt.dll
import nume_functie3 msvcrt.dll
```

1.b. Convenții de apel

O *convenție de apel (a calling convention)* definește un ansamblu de reguli referitoare la:

– modul și ordinea de transmitere a argumentelor funcției;

– unde va fi returnat rezultatul execuției;

– care sunt regiștrii care pot fi modificați pe timpul execuției;

– cine are obligația să *„șteargă”* argumentele transmise.

Apelul unei funcții din biblioteca *msvcrt.dll (Microsoft Visual C Runtime)* necesită utilizarea convenției de apel `_cdecl`. Această convenție de apel specifică următoarele reguli:

– argumentele funcției apelate se pun pe stivă începând cu ultimul argument din lista de argumente (de la dreapta spre stânga listei);

– dacă funcția apelată returnează un rezultat, acesta va fi depus implicit în registrul EAX;

– funcția apelată nu va ***„șterge”*** argumentele puse pe stivă (este sarcina funcției apelante);

– pe timpul execuției funcției apelate conținutul regiștrilor EAX, ECX și EDX poate fi modificat.

1.c. Funcții pentru afișare pe ecran

Pentru afișarea rezultatelor pe ecran (la consolă) vom utiliza funcția:

```
printf(const char* format, val_1, val_2, ...)
```

unde:

`format` este offset-ul șirului de caractere care specifică formatul de afișare a datelor, astfel:

Specificator	Ce se afișează	Exemplu	Dimensiune de reprezentare
`"%c"`	Caracter	***a***	octet
`"%d"` **`"%i"`**	Număr cu semn în baza 10	***392*** ***–1024***	dublucuvânt
`"%u"`	Număr fără semn în baza 10	***7235***	dublucuvânt
`"%x"`	Număr fără semn în baza 16	***7fa***	dublucuvânt
`"%s"`	Șir de caractere (terminat cu 0)	***"ana are mere"***	șir de octeți terminat cu 0

1.d. Funcții pentru citirea de la tastatură

Pentru citirea datelor de la tastatură putem utiliza funcțiile:

```
gets(char* buffer)
scanf(const char* format [, argument]...)
```

unde:

`buffer` este offset-ul zonei de memorie în care vor fi citite datele

`format` este specificatorul de format (vezi tabelul de mai sus)

2. OPERAȚII CU FIȘIERE DE TIP TEXT

Lucrul cu fișiere implică execuția următoarelor operații:

a. deschiderea/crearea fișierului;

b. prelucrarea datelor (citirea și/sau scrierea datelor din/în fișier);

c. închiderea fișierului.

2.a. Deschiderea/Crearea unui fișier text

Pentru deschiderea/crearea unui fișier de tip text vom utiliza funcția `fopen()`.

Prototipul acestei funcții în limbajul C este:

```
FILE* fopen(const char* nume_fisier, const char* mod_acces)
```

unde:

`nume_fisier` este offset-ul unui șir de caractere care specifică numele fișierului (calea absolută/relativă)

`mod_acces` este offset-ul unui șir de caractere care specifică drepturile de acces, astfel:

Mod acces	Drepturi de acces	Descriere
`"r"`	citire (read)	Deschide un fișier text pentru citire. Fișierul trebuie să existe pe disc. Dacă acest fișier nu există, funcția va returna valoarea 0 (eroare la deschiderea fișierului).
`"w"`	scriere (write)	Dacă nu există un fișier cu acel nume, va crea fișierul și îl va deschide pentru scriere. Dacă există deja un fișier cu acel nume, îl va deschide pentru scriere, însă conținutul inițial al fișierului va fi suprascris (scrierea se va face de la începutul fișierului).
`"a"`	adăugare (append)	Dacă nu există un fișier cu acel nume, va crea fișierul și îl va deschide pentru scriere. Dacă există deja un fișier cu acel nume, îl va deschide pentru scriere, însă conținutul inițial al fișierului va fi păstrat (scrierea se va face de la sfârșitul fișierului).
`"r+"`	citire și scriere	Deschide un fișier text pentru citire și scriere. Fișierul trebuie să existe pe disc. Dacă acest fișier nu există, funcția va returna valoarea 0 (eroare la deschiderea fișierului).
`"w+"`	citire și scriere	Dacă nu există un fișier cu acel nume, va crea fișierul și îl va deschide pentru citire și scriere. Dacă există deja un fișier cu acel nume, îl va deschide pentru citire și scriere, însă conținutul inițial al fișierului va fi suprascris (scrierea se va face de la începutul fișierului).
`"a+"`	citire și adăugare	Dacă nu există un fișier cu acel nume, va crea fișierul și îl va deschide pentru citire și scriere. Dacă există deja un fișier cu acel nume, îl va deschide pentru citire și scriere, însă conținutul inițial al fișierului va fi păstrat (scrierea se va face de la sfârșitul fișierului).

Dacă apelul funcției s-a realizat cu succes, `fopen()` va returna un descriptor de fișier (un pointer către structura `FILE`). Dacă apelul a eșuat, funcția va returna valoarea 0.

Observații

1. Este NECESAR să stocăm *descriptorul de fișier* într-o variabilă definită în segmentul de date, deoarece acest descriptor va fi folosit de către funcțiile de citire/scriere din/în fișier.

2. Având în vedere că există situații în care apelul funcției `fopen()` poate să eșueze, este OBLIGATORIU să verificăm dacă fișierul a fost deschis cu succes sau nu:

```
cmp eax, 0
je eroare_deschidere_fisier
```

2.b.1. Citirea dintr-un fișier text

Citirea dintr-un fișier care a fost deschis anterior cu `fopen()` se poate realiza cu ajutorul funcțiilor:

```
int fread(void* buffer, int size, int count, FILE* stream)
int fscanf(FILE* stream, const char* format, ...)
```

unde:

`buffer` este offset-ul zonei de memorie în care vor fi citite datele

`size` este dimensiunea tipului de date care vor fi citite

`count` este numărul de elemente care vor fi citite

`stream` este descriptorul de fișier (un pointer către structura `FILE`)

`format` specifică formatul operației de citire din fișier

2.b.2. Scrierea într-un fișier text

Scrierea într-un fișier care a fost deschis anterior cu `fopen()` se poate realiza cu ajutorul funcțiilor:

```
int fwrite(void* buffer, int size, int count, FILE* stream)
```

```
int fprintf(FILE* stream, const char* format, ...)
```

unde:

`buffer` este offset-ul zonei de memorie din care vor fi scrise datele

`size` este dimensiunea tipului de date care vor fi scrise

`count` este numărul de elemente care vor fi scrise

`stream` este descriptorul de fișier (un pointer către structura `FILE`)

`format` specifică formatul operației de scriere în fișier

2.c. Închiderea unui fișier text

Închiderea unui fișier care a fost deschis anterior cu `fopen()` se realizează în mod obligatoriu cu ajutorul funcției:

```
int fclose(FILE* descriptor_fisier)
```

unde:

`descriptor_fisier` este descriptorul de fișier (un pointer către structura `FILE`) returnat după execuția cu succes a apelului funcției `fopen()`

EXERCIȚII

1. Scrieți un program care citește de la tastatură un număr întreg și îl afișează în baza 2.

2. Scrieți un program care citește de la tastatură un număr întreg și afișează câte cifre are numărul citit în baza 10.

3. Scrieți un program care citește de la tastatură un șir de numere întregi și îl afișează în ordine crescătoare/descrescătoare.

4. Scrieți un program care citește de la tastatură cuvinte (șiruri de caractere), verifică și, apoi, afișează dacă aceste cuvinte sunt de tip palindrom.

5. Se dă un nume de fișier (definit în segmentul de date). Fișierul conține numerele întregi cu semn separate prin spații. Să se calculeze și să se afișeze suma numerelor din fișier.

6. Se dă un nume de fișier (definit în segmentul de date). Fișierul conține litere mici, litere mari, cifre și caractere speciale. Să se înlocuiască toate caracterele speciale din fișier cu caracterul 'X'.

7. Se dă un nume de fișier (definit în segmentul de date). Fișierul conține cuvinte (șiruri de caractere separate prin spații). Să se determine și să se afișeze numărul de cuvinte din fișier.

8. Se dă un nume de fișier (definit în segmentul de date). Fișierul conține propoziții (șiruri de caractere separate prin spații sau punct). Să se determine și să se afișeze numărul de propoziții din fișier.