```
#include <iostream>
using namespace std;

int exercitiul1_implementare(int n)
{
	_asm {
		mov eax, [ebp + 8]; //iau primul parametru
		mov ecx, 0; //contorul
		mov ebx, 0; //suma
	_for:
		cmp ecx, eax; //compar contorul cu n
		jg _final; //daca contor > n sar la final
		add ebx, ecx; //fac in ebx suma
		inc ecx; //incrementez ecx
		jmp _for;
	_final:
		mov eax, ebx; //pun in eax ce vreau sa returnez
	}
}
void exercitiul1()
{
	//citesc n
	int n, result;
	cout << "Introduceti n: ";
	cin >> n;
	_asm {
		mov eax, n;
		push eax;
		call exercitiul1_implementare;
		mov result, eax;
		add esp, 4; //adaug la esp 4 pentru ca am dat push doar la un parametru
	}
	cout << "Rezultatul este: " << result << "\n";
}


int exercitiul2_implementare(int n)
{
	_asm {
		mov eax, [ebp + 8]; //iau primul parametru
		mov ebx, 10; //la ce vreau sa impart
		push 0; //fac o variabila locala in care voi retine suma
	_for:
		cmp eax, 0; //compar eax, cu 0
		je _final; //daca eax e 0, am sarit
		mov edx, 0; //ma pregatesc de impartire, fac edx 0
		div ebx; //fac div, in eax o sa am eax / 10, in edx o sa am eax % 10
		add [esp], edx; //adaug la variabila de pe stiva ultima cifra
		jmp _for;
	_final:
		mov eax, [esp]; //pun in eax variabila locala
		add esp, 4; //refac stiva, scot variabila locala
	}
}
void exercitiul2()
{
	//citesc n
	int n, result;
	cout << "Introduceti n: ";
	cin >> n;
```

```
	_asm {
		mov eax, n;
		push eax;
		call exercitiul2_implementare;
		mov result, eax;
		add esp, 4; //adaug la esp 4 pentru ca am dat push doar la un parametru
	}
	cout << "Rezultatul este: " << result << "\n";
}


int exercitiul3_implementare(int n)
{
	_asm {
		mov eax, [ebp + 8]; //iau primul parametru
		mov ebx, 10; //la ce vreau sa impart
		push 0; //fac o variabila locala in care voi retine suma
	_for:
		cmp eax, 0; //compar eax, cu 0
		je _final; //daca eax e 0, am sarit
		mov edx, 0; //ma pregatesc de impartire, fac edx 0
		mov ebx, 10; //fac ebx-ul cu 10
		div ebx; //fac div, in eax o sa am eax / 10, in edx o sa am eax % 10
		mov ecx, edx; //salvez ultima cifra intr-un nou registru
		push eax; //salvez eax, (numarul meu / 10)
		mov edx, 0;
		mov eax, ecx;
		mov ebx, 2;
		div ebx;
		pop eax; //recapat eax-ul
		cmp edx, 1; //compar restul cu 1;
		je _for; //daca cifra e impara, nu mai fac suma
		add [esp], ecx; //adaug la variabila de pe stiva ultima cifra pe care am
retinut-o in ecx
		jmp _for;
	_final:
		mov eax, [esp]; //pun in eax variabila locala
		add esp, 4; //refac stiva, scot variabila locala
	}
}
void exercitiul3()
{
	//citesc n
	int n, result;
	cout << "Introduceti n: ";
	cin >> n;
	_asm {
		mov eax, n;
		push eax;
		call exercitiul3_implementare;
		mov result, eax;
		add esp, 4; //adaug la esp 4 pentru ca am dat push doar la un parametru
	}
	cout << "Rezultatul este: " << result << "\n";
}


void exercitiul4_implementare(int *, int)
{
	_asm {
```

```
		mov eax, [ebp + 8]; //iau primul parametru, vectorul
		mov ebx, [ebp + 12]; //iau size-ul lui
		mov ecx, 0; //primul contor 0
		mov edx, 0; //al doilea contor 0
	_first_for:
		cmp ecx, ebx; //compar primul contor cu lungime sir, daca e >= ies
		jge _outside_first_for;
		mov edx, ecx;
		inc edx; //fac edx i + 1
	_second_for:
		cmp edx, ebx;
		jge _outside_second_for;
		//cmp v[i], cu v[j] si daca e mai mare, fac swap
		mov edi, [eax + ecx * 4]; //(ecx * 4) pentru ca sizeof(int) = 4
		mov esi, [eax + edx * 4];
		cmp edi, esi;
		jle _no_swap; //daca e mai mic sau egal nu fac swap
		xchg edi, esi;
		mov [eax + ecx * 4], edi;
		mov [eax + edx * 4], esi;
	_no_swap:
		inc edx;
		jmp _second_for;
	_outside_second_for:
		inc ecx; //fac i++ //continui forul
		jmp _first_for;
	_outside_first_for:
	}
}
void exercitiul4()
{
	int v[] = { 6, 8, 9, 1, 2, 5, 3, 10 };
	int dimensiune_vector = 8;
	_asm {
		mov eax, dimensiune_vector; //pun in eax dimensiunea vectorului (dau parametrii
in ordine inversa)
		push eax;
		lea eax, v; //incarc in eax adresa lui v
		push eax;
		call exercitiul4_implementare;
		add esp, 8; //adaug la esp 8 pentru ca am dat push la doi parametri
	}
	cout << "Vectorul rezultat: ";
	for (int i = 0; i < dimensiune_vector; i++)
	{
		cout << v[i] << " ";
	}
	cout << "\n";
}


int exercitiul5_implementare(int)
{
	_asm {
		mov ebx, [ebp + 8]; //iau primul parametru, n
		cmp ebx, 1;//daca e 1, returnez 1
		je _return_value_1;
		mov ecx, ebx; //in ecx voi retine produsul
		dec ebx; //decrementez ebx
		push ecx; //salvez ecx-ul
```

```
            push ebx; //trimit un parametru
            call exercitiul5_implementare; //fac (ebx-1)! factorial
            add esp, 4; //curat stiva de parametrul trimisi
            pop ecx; //recuperez ecx-ul;
            mul ecx; //in eax am valoarea returnata, pe care o inmultesc cu n (ceva de
genul: n * (n-1)!)
            jmp _final; //sar peste return 1;
      _return_value_1:
            mov eax, 1;
      _final:

      }
}
void exercitiul5()
{
      int n, rezult;
      cout << "Introduceti n: " << "\n";
      cin >> n;
      _asm {
            mov eax, n;
            push eax;
            call exercitiul5_implementare;
            add esp, 4; //adaug la esp 4 pentru ca am dat push la un singur parametru
            mov rezult, eax;
      }
      cout << "Rezultatul lui " << n << "!: " << rezult;
      cout << "\n";
}


void exercitiul6_implementare(char * /* sir */ , int /*size sir*/, int /*caracter de
cautat*/ )
{
      _asm {
            mov eax, [ebp + 8]; //iau primul parametru, sirul
            mov ebx, [ebp + 12]; //iau size-ul lui
            mov edx, [ebp + 16]; //caracterul de cautat
            mov ecx, 0;
            push 0; //fac o variabila locala in care voi retine numarul de caractere
cautate
      _for:
            cmp ecx, ebx;
            jge _final;
            movzx esi, byte ptr [eax + ecx]; //pun in esi valoarea de la adresa [eax + ecx]
            //folosesc byte ptr pentru a preciza clar ca vreau doar un caracter
            //folosesc movzx pentru a putea copia valoarea in esi (esi e pe 4, caracterul e
1, ar da operand size conflict)
            cmp esi, edx;
            jne _nu_numara;
            add [esp], 1; //inseamna ca esi == edx, deci pot sa numar
      _nu_numara:
            inc ecx;
            jmp _for;
      _final:
            mov eax, [esp]; // iau variabila locala
            add esp, 4; //refac stiva. am folosit o singura variabila locala, adaug 4
      }
}
void exercitiul6()
{
```

```
	char c[] = "ana are mere";
	int size_sir = 12;
	char to_find = 'a';
	int result;
	_asm {
		movzx eax, to_find; //char are size 1, ca sa-l pun in eax, trebuie sa folosesc
movzx
		push eax;
		mov eax, size_sir;
		push eax;
		lea eax, c; //incarc in eax adresa lui c
		push eax;
		call exercitiul6_implementare;
		add esp, 12; //adaug la esp 12 pentru ca am dat push la trei parametri
		mov result, eax;
	}
	cout << "Valoarea rezultata: ";
	cout << result;
	cout << "\n";
}


void exercitiul7_implementare(int * /* matrice A */, int * /* matrice B */, int /* size
matrice*/)
{
	_asm {
		mov ebx, [ebp + 8]; //iau prima matrice
		mov ecx, [ebp + 12]; //iau a doua matrice
		mov esi, [ebp + 16]; //iau size-ul
		push 4; //sizeof(int)
		push 0; //initializez contor 2 cu 0
		push 0; //initializez contor 1 cu 0
	_for1:
		cmp [esp], esi;
		jge _outside_for_1;
		mov [esp + 4], 0; //fac j-ul 0
	_for2:
		cmp [esp + 4], esi;
		jge _outside_for_2;

		//elementul A[i][j] se va afla la A + i * numarul de coloane * sizeof(int) + j
* sizeof(int)
		mov eax, [esp];
		mul [esp + 8];
		mul esi;
		mov edi, eax; //salvez in eax i * numarul de coloane * sizeof(int)
		mov eax, [esp + 4];
		mul [esp + 8];
		add edi, eax; //obtin in edi: i * numarul de coloane * sizeof(int) + j *
sizeof(int);
		mov eax, [ebx + edi];
		add eax, [ecx + edi]; // A[i][j] += B[i][j]
		mov [ebx + edi], eax;
		inc [esp + 4]; //incrementez j-ul
		jmp _for2;
	_outside_for_2:
		inc [esp]; //incrementez i
		jmp _for1;
	_outside_for_1:
		add esp, 12; //am avut 3 variabile locale
```

```
	}
}
void exercitiul7()
{
	int n = 3;
	int A[3][3] = { {1,0,1},{0,1,0},{ 1,0,1 } };
	int B[3][3] = { {2,1,2},{1,2,1},{ 2,1,2 } };
	_asm {
		mov eax, n;
		push eax;
		lea eax, B;
		push eax;
		lea eax, A;
		push eax;
		call exercitiul7_implementare;
		add esp, 12; //adaug la esp 12 pentru ca am dat push la trei parametri
	}
	cout << "Matricea finala A: \n";
	for (int i = 0; i < n; i++) {
		for (int j = 0; j < n; j++)
			cout << A[i][j] << " ";
		cout << "\n";
	}
	cout << "\n";
}


struct Point {
	int x, y;
};
void exercitiul8_implementare(Point *, Point*)
{
	_asm {
		mov esi, [ebp + 8];
		mov edi, [ebp + 12];
		mov ecx, 0;
		mov edx, 0;
		mov ecx, [edi];
		sub ecx, [esi];
		mov eax, ecx;
		mul eax;
		mov ecx, eax;

		mov edx, [edi + 4];
		sub edx, [esi + 4];
		mov eax, edx;
		mul eax;
		mov edx, eax;

		add ecx, edx;
		mov eax, ecx;
	}
}
void exercitiul8()
{
	Point A;
	Point B;
	A.x = 3; A.y = 4;
	B.x = 5; B.y = 20;
	int rezultat;
```

```
	_asm {
		lea eax, B;
		push eax;
		lea eax, A;
		push eax;
		call exercitiul8_implementare;
		add esp, 8; //adaug la esp 8 pentru ca am dat push la 8 parametri
		mov rezultat, eax;
	}
	cout << A.x << " " << A.y << "\n";
	cout << B.x << " " << B.y << "\n";
	cout << "Patratul distantei: " << rezultat << "\n";
}


int main()
{
	//recapitulare laborator: https://profs.info.uaic.ro/~rvlad/lab/acso/labs-acso.pdf

	//Exercitiul 1.
	//Scrieti o functie care face suma numerelor de la 1 la n. s = 1 + 2 + 3 + ... + n
	//pentru n = 10, vom obtine 55
	//exercitiul1();

	//Exercitiul 2.
	//Scrieti o functie care face suma cifrelor unui numar n (vom retine suma intr-o
variabila locala)
	//pentru n = 1234, vom obtine 10
	//exercitiul2();

	//Exercitiul 3.
	//Scrieti o functie care face suma cifrelor pare ale unui numar n (putem retine suma
intr-o variabila locala)
	//pentru n = 128879, vom obtine 18
	//exercitiul3();


	//Exercitiul 4.
	//Scrieti o functie sa sorteze crescator un vector dat.
	//pentru vectorul v: 6 8 9 1 2 5 3 10, vom obtine 1 2 3 5 6 8 9 10
	//functia va lua ca parametri vectorul si size-ul acestuia
	//exercitiul4();


	//Exercitiul 5
	//Sa se calculeze factorialul unui numar dat n
	//pentru n = 5, rezultatul e 120
	//exercitiul5();


	//Exercitiul 6
	//Scrieti o functie sa numere de cate ori un caracter dat apare intr-un sir.
	//pentru sirul: "ana are mere" si caracterul "a" se va return 3
	//functia va lua ca parametri sirul si caracterul (il vom trimite ca int) si size-ul
sirului
	//exercitiul6();


	//Exercitiul 7
```

```
    //Fiind date 2 matrici patratice alocate static cu aceeasi dimensiune cu elemente de
tip int, A si B, sa se calculeze in A, A+B (A = A+B) si sa se afiseze A.
    /*
    pentru A:
    1 0 1
    0 1 0
    1 0 1
    si B:
    2 1 2
    1 2 1
    2 1 2

    valoarea finala a lui A va fi:
    3 1 3
    1 3 1
    3 1 3
    */
    //functia va lua ca parametri cele 2 matrici si dimensiunea lor
    //exercitiul7();


    //Exercitiul 8
    //Fiind date 2 Puncte A si B (un punct e definit de coordonatele x si y) sa se
returneze patratul distantei intre cele 2 puncte
    //exercitiul8();

    system("pause"); //pun system("pause") doar daca sunt pe empty project
    return 0;
}
```