```
void Exercicio1();//inverte os valores de A e B

void Exercicio2();// Somar o conteúdo de um array

void Exercicio3();// Dados dos números A e B, se A > B então Variável C = 1, caso contrário C = 0;

void Exercicio4();// Dados os termos da equação de bascará: A,B e C calcule o Delta

void Exercicio5();// Dada uma tabela chamada Tabela1 com 10 elementos quaisquer grave na variável R maior valor
da tabela.

void Exercicio6();// Descobrir tamanho de uma string

void Exercicio7();// Dada duas variáveis strings cujo a primeira chama se STR1 e esta com o valor “FACENS” cujo o
tamanho é 6 caracteres.Faça a transferência de dados de STR1 para STR2.

void Exercicio8();// Dado duas strings STR1 e STR2 faça a comparação das mesmas, se STR1 = STR2 então Variável R
= 1 caso contrário R = 0

void Exercicio9();// Data uma tabela chamada TabelaSort com 10 elementos, efetua a ordenação da mesma.(utilize o
algoritmo bolha)

int main() {

	Exercicio1();
	Exercicio2();
	Exercicio3();
	Exercicio4();
	Exercicio5();
	Exercicio6();
	Exercicio7();
	Exercicio8();
	Exercicio9();

	getchar();

	return 0;
}

void Exercicio9()
{
	int v[] = { 9, 10, 8, 7, 6, 5, 4, 3, 2, 1 };
	int aux, i, flag = 1, tam = 10;

	__asm
	{
	inicio:
		lea eax, v
			mov ebx, flag
			cmp ebx, 1
			jne fim
			mov ebx, 0
			mov flag, ebx

			mov ecx, 0
			innerFor:
		mov i, ecx
			mov edx, tam
			add edx, -1
			cmp ecx, edx
			jge inicio
			mov edx, [eax + 4]
			cmp edx, [eax]
			jge incremento; Mudar aqui para jle para trocar entre crescente / decrescente
			mov ecx, [eax]
			mov[eax], edx
			mov[eax + 4], ecx
			mov ebx, 1
			mov flag, ebx
			incremento :
		mov ecx, i
			inc ecx
			add eax, 4
			jmp innerFor
```

```
                fim :
        }

        for (int i = 0; i < 10; i++)
        {
                cout << v[i] << " ";
        }
        cout << endl;
}

void Exercicio8()
{
        cout << "Exercicio 8" << endl;

        char s[7] = "FACENS";
        char s2[7] = "FACENS";
        int R;

        __asm
        {
                lea eax, s
                lea ebx, s2
                mov ecx, 1
                mov R, ecx

                looping :
                cmp[eax], '\0'
                        je fim
                        mov ecx, [eax]
                        cmp ecx, [ebx]
                        jne naoiguais
                        add eax, 1
                        add ebx, 1
                        jmp looping

                        naoiguais :
                mov ecx, 0
                        mov R, ecx
                        jmp fim

                        fim :
        }

        cout << R << endl;
}

void Exercicio7()
{
        cout << "Exercicio 7" << endl;

        char s[7] = "FACENS";
        char s2[7] = "       ";

        __asm
        {
                lea eax, s
                lea ebx, s2

                looping :
                cmp[eax], '\0'
                        je fim
                        mov ecx, [eax]
                        mov[ebx], ecx
                        add ebx, 1
                        add eax, 1
                        jmp looping

                        fim :
        }

        cout << s2 << endl;
}

void Exercicio6()
{
```

```
	cout << "Exercicio 6" << endl;

	int tam;

	char s[] = "FACENSSSSSSSSSS";

	__asm
	{
		lea eax, s
		xor ebx, ebx
		xor ecx, ecx
		mov tam, ecx

		looping :
		cmp[eax], '\0'
			je fim
			add ecx, 1
			add eax, 1
			jmp looping

			fim :
		mov tam, ecx
	}

	cout << tam << endl;
}

void Exercicio5()
{
	cout << "Exercicio 5" << endl;

	int v[] = { 10, 2, 3, 4, -1, 6, 7, 8, 9, 10 };
	int R, menor;

	__asm
	{
		lea eax, v
		xor cx, cx
		mov edx, [eax]
		mov menor, edx
		add eax, 4

		looping:
		cmp cx, 9
			je fim
			mov edx, menor
			mov ebx, [eax]
			cmp[eax], edx
			jl troca
			jmp proximo
			troca :
		mov edx, [eax]
			mov menor, edx
			proximo :
		inc cx
			add eax, 4
			jmp looping

			fim :
		mov eax, menor
			mov R, eax
	}

	cout << R << endl;
}

void Exercicio4()
{
	cout << "Exercicio 4" << endl;
	int A, B, C, delta, aux;
	A = 1;
	B = 4;
	C = 3;

	__asm
```

```
	{
		// Resultado do mul fica no EAX
		mov eax, B
		mul eax
		mov delta, eax

		mov eax, -4
		imul A
		imul C

		mov ebx, delta
		add eax, ebx
		mov delta, eax
	}

	cout << delta << endl;
}

void Exercicio3()
{
	cout << "Exercicio 3" << endl;
	int A, B, C;
	A = 3;
	B = 2;

	__asm
	{
		mov eax, A
		mov ebx, B
		mov ecx, C
		cmp eax, ebx
		jg cum
		jmp czero

		cum :
		mov ecx, 1
			mov C, ecx
			jmp fim
		czero :
		mov ecx, 0
			mov C, ecx

			fim :
	}

	cout << C << endl;

}

void Exercicio2()
{
	cout << "Exercicio 2" << endl;
	int vet[] = { 1, 5, 2, 6, 4, 1, 5, 2, 6, 4 }, s, aux;

	__asm
	{
		mov eax, 0
		mov s, eax
		xor cx, cx
		lea eax, vet

		looping :
		cmp cx, 10
			jge fim
			mov edx, [eax]
			mov aux, edx
			mov edx, s
			add edx, aux
			mov s, edx
			add eax, 4
			inc cx
			jmp looping

			fim :
	}
```

```
	cout << s << endl;
}

void Exercicio1()
{
	cout << "Exercicio 1" << endl;

	int a = 5, b = 10;

	__asm
	{
		mov eax, a
		mov ecx, b
		push eax
		push ecx
		call troca
		pop b
		pop a
		jmp fim

		troca :
		push ebp
			mov ebp, esp
			push ebx
			push esi
			push edi

			mov eax, [ebp + 12]
			mov ecx, [ebp + 8]
			mov[ebp + 12], ecx
			mov[ebp + 8], eax

			pop edi
			pop esi
			pop ebx
			pop ebp
			ret
			fim :

	}

	printf("%d %d\n", a, b);

}


void inverterString()
{
	cout << "Inverter" << endl;

	char s[7] = "FACENS";
	char s2[7] = "FACENS";

	__asm
	{
		lea ecx, s
		lea ebx, s
		lea edx, s2

		looping :
		cmp[ecx], '\0'
			je looping2
			mov eax,0
			mov al, [ecx]
			push al
			add ecx, 1
			jmp looping

			looping2 :
		cmp[edx], '\0'
			je fim
			pop ecx
			mov[ebx], ecx
			add ebx, 1
```

```
                add edx, 1
                jmp looping2
                fim :

        }

        cout << s << endl;

}


        cout << "Exercicio 8" << endl;

        int vet[] = { 5,4,3,2,1 };

        int i, tam;

/*

conta_tam:

        cmp[eax], '\0'

                je inicio

                add tam, 1

                add eax, 1

                jmp conta_tam*/


        __asm

        {

        inicio:

                lea eax, vet

                        mov i, 0

                        pushando :

                        cmp i, 5

                        jge lala

                        mov ebx, [eax]

                        push ebx

                        add eax, 4

                        mov ecx, i

                        inc ecx

                        mov i, ecx

                        jmp pushando


                        lala :

                lea eax, vet

                        mov i, 0


                        popando :

                        cmp i, 5

                        jge fim

                        pop ebx

                        mov[eax], ebx
```

```
                add eax, 4
                mov ecx, i
                inc ecx
                mov i, ecx
                jmp popando

                fim :
}
```