Laboratorium programowania niskopoziomowego

LAB 8 – Operacje na łańcuchach.

Łańcuchy znaków *Char** O P I S

Łańcuch znaków *Char** to tablica jednowymiarowa znaków zapisanych w kodzie ASCI. Na przykład napis w nawiasach kwadratowych: [Ala ma kota] wraz ze spacjami zostanie zapisany w następujący sposób w pamięci operacyjnej:

Adres w pamięci	**HEX**	**ASCII**
0x008E79C8	41	A
0x008E79C9	6c	l
0x008E79CA	61	a
0x008E79CB	20	
0x008E79CC	6d	m
0x008E79CD	61	a
0x008E79CE	20	
0x008E79CF	6b	k
0x008E79D0	6f	o
0x008E79D1	74	t
0x008E79D2	61	a
0x008E79D3	20	
0x008E79D4	20	
0x008E79D5	20	
0x008E79D6	20	
0x008E79D7	20	
0x008E79D8	20	
0x008E79D9	20	
0x008E79DA	20	
0x008E79DB	20	
0x008E79DC	20	
0x008E79DD	20	
0x008E79DE	20	
0x008E79DF	20	
0x008E79E0	20	
0x008E79E1	20	
0x008E79E2	20	
0x008E79E3	20	
0x008E79E4	20	
0x008E79E5	20	
0x008E79E6	20	
0x008E79E7	20	
0x008E79E8	00	.

Proszę zauważyć, że pod adresem 0x008E79E8 został zapisany znak [.] o kodzie heksadecymalnym wynoszącym „00”. Jest to dodatkowa informacja, że tu następuje koniec łańcucha znaków. Informacja ta zostanie wykorzystana w dalszej części laboratorium.

Zad 1. Bazując na powyższym proszę przeanalizować kod C++:

```
int size_cpp(char *tab)
{
	int rozmiar = 0;
	while (tab[rozmiar++]);
	return --rozmiar;
}
```

Oraz kod asm:

```
int size_asm(char *tab, int n)
{
	__asm {
		pushfd
		xor eax, eax;		//eax <- 0 (NULL)
		mov edi, tab;		//początek tablicy
		mov ecx, n;		//liczba elementów
		cld;			//flaga kierunku danych
		repne scasb;		//powtarzaj dopóki nierówne porównanie al i [edi]
					//(edi++, ecx--) tab[rozmiar]!=NULL;
		mov eax, n;		//ponownie wielkość tablicy
		sub eax, ecx;		//(N - (N – (rozmiar +1)))
		dec eax;		//(rozmiar + 1 -1)
		popfd
	}
}
```

Przykładowy kod funkcji *main*:

```
int main()
{
	const int n = 100;
	int start = 2;
	int ile = 7;
	char tab[n] = "Ala ma kota                    ";
	char tab2[n] = "";
	char tab3[n] = "";

	cout << "size_cpp " << size_cpp(tab) << endl;
	cout << "size_asm " << size_asm(tab, n) << endl << endl;
	cout << "find_cpp " << find_cpp(tab, 'k') << endl;
	cout << "find_asm " << find_asm(tab, n, 'k') << endl << endl;

	cout << "[" << tab << "]" << endl;
	cout << "r_trim_cpp [" << r_trim_cpp(tab) << "]" << endl << endl;

	memcpy(tab, "Ala ma kota                    ",32);
	cout << "[" << tab << "]" << endl;
	cout << "r_trim_asm [" << r_trim_asm(tab, n) << "]" << endl << endl;

	memcpy(tab, "Ala ma kota                    ",32);
	cout << "[" << tab2 << "]" << endl;
	cout << "substr_cpp [" << substr_asm(tab, tab2, start, ile) << "]" << endl;
	cout << "[" << tab3 << "]" << endl;
	cout << "substr_asm [" << substr_asm(tab, tab2, start, ile) << "]" << endl << endl;

	system("PAUSE");
	return 0;
}
```

UWAGA: funkcja pomocnicza **memcpy()** kopiuje znaki do tablicy znaków.

Zad 2 Proszę przeanalizować program zwracający numer indeksu pierwszego wystąpienia danego znaku:

Kod C++:

```
int find_cpp(char * tab, char znak)
{
	int n = 0, poz;
	while (tab[n])
		if (tab[n++] == znak)
		{
			poz = n;
			break;
		}
	return --poz;
}
```

Kod asm:

```
int find_asm(char * tab, int n, char znak)
{
	int poz;
	__asm {
		pushfd;
		xor eax, eax;
		mov al, znak;		//eax <- 0 (NULL)
		mov edi, tab;		//początek tablicy
		mov ecx, n;	//liczba elementów
		cld;			//kasowanie flagi kierunku. Ustaw (std) flagę (df) edi++
		repne scasb;		//powtarzaj dopóki nierówne porównanie al i [edi]
					//(edi++, ecx--) tab[rozmiar]!=NULL;
		mov eax, n;	//ponownie wielkość tablicy
		sub eax, ecx;	//(N - (N – (rozmiar + 1)))
		dec eax;		//(rozmiar + 1 -1)
		mov poz, eax;
		popfd;


	}
	return poz;
}
```

Zad3. Proszę przeanalizować kod odpowiedzialny za eliminację prawych spacji z łańcucha znaków. Proszę opowiedzieć co się stało z łańcuchem znaków po wykonaniu funkcji C++ i asm

Kod C++:

```
char * r_trim_cpp(char * tab)
{
	int N = size_cpp(tab);
	N--;
	while (tab[N])
	{
		if (tab[N] != ' ') //znajdź pierwszy znak różny od spacji
		{
			N++;
			tab[N] = '\0';
			break;
		}
		N--;
	}
	return tab;
}
```

Kod asm:

```
char * r_trim_asm(char * tab, int n)
{
	__asm
	{
		pushfd
		xor eax, eax;
		mov edi, tab;
		mov ecx, n;	//liczba elementów (max)
		cld;		//flaga kierunku danych
		repne scasb;	//powtarzaj dopóki nierówne porównanie al i [edi]
		sub edi, 2;	//ostatni element tekstu
		std;
		mov ecx, n;
		mov al, 32;
		repe scasb;
		add edi, 2;
		mov[edi], 00h;
		popfd;
	}
	return tab;
}
```

Zad4. Napisz samodzielnie funkcję C++ i asm kopiującą łańcuch znaków z Tab1 do Tab2.

Zad5. Napisz samodzielnie funkcję C++ i asm kopiującą łańcuch znaków z Tab1 od indeksu X do indeksu Y do Tab2.

Zad6. Napisz samodzielnie funkcję C++ i asm usuwającą lewe spacje z łańcucha znaków .