Programowanie niskopoziomowe

Ćwiczenia laboratoryjne w środowisku Visual Studio 2019

Labolatoria 3

„Przesyłanie danych i zarządzanie danymi”

1 Zadania

Wskazówki do zadań 1.1-1.4 w dokumencie „lab03 pn Materiały uzupełniające.pdf”.

1.1 Kopiowanie łańcuchów

Przykład lab03zad1.asm zmodyfikuj tak, aby skopiować zawartość bufora do bufora1. Wyświetl zawartość bufora1.

Wskazówka: skorzystaj z instrukcji łańcuchowej MOVSB w połączeniu z REP

1.2 Przesyłanie danych przez stos

Korzystając ze stosu odwróć kolejność znaków w buforze1. Ponownie wyświetl zawartość bufora.

1.3 Przesyłanie danych z pamięci do akumulatora

Wczytaj dziesięć liczb podanych przez użytkownika do tablicy bajtowej (skorzystaj z przykładu lab03zad3.asm). Oblicz sumę tych liczb.

Wskazówka: skorzystaj z instrukcji łańcuchowej LODSB, obliczeń dokonaj w stosownej pętli.

1.4 Przesyłanie danych z pamięci do akumulatora

Wczytaj dziesięć liczb podanych przez użytkownika do tablicy bajtowej. Następnie wyświetl sumę liczb wybranych wg swojego wariantu (prowadzący określa co którą wprowadzoną liczbę dodajemy).

1.5 Tryby adresowania

W tekście ostatniego programu wyszukaj przykłady różnych trybów adresowania i wskaż stosowany tryb w komentarzu do instrukcji.

Tryby adresowania:

1. Adresowanie rejestrowe
2. Adresowanie natychmiastowe
3. Adresowanie bezpośrednie pamięci
4. Adresowanie bezpośrednie z przesunięciem
5. Adresowanie pamięci pośrednie

Wskazówki do zadanie 1.5

Adresowanie rejestrowe:

```
mov EAX, 4          ;rejestr jako pierwszy parametr
mov rozmA, EAX      ;rejestr jako drugi parametr
mov EBX,EAX         ;rejstry w obu parametrach
```

Adresowanie natychmiastowe, warto zwrócić uwagę że rozmiar stałej jest zależy od pierwszego operatora:

```
byte_val      DB     123    ;stała 123 zalokowana jest w odpowiednim miejscu w pamięci
add byte_val, 65            ;stała 65 dodana jest do zmiennej byte_val
mov AX, 45h                 ;stała zapisana jest do rejestru AX
```

Adresowanie bezpośrednie pamięci:

```
add byte_value, DL   ;dodanie rejestru do wartości w pamięci
mov BX, word_value   ;operand z pamięci dodany do rejestru
```

Adresowanie bezpośrednie z przesunięciem, zakładamy że byte_table oraz word_table to tablice w pamięci:

```
mov CL, byte_table[2]       ; pobranie do rejestru cl trzeciego elementu tablicy
mov CL, byte_table + 2      ; pobranie do rejestru cl trzeciego elementu tablicy
mov CX, word_table[3]       ; pobranie do rejestru cx czwartego elementu tablicy
mov CX, word_table + 3      ; pobranie do rejestru cx czwartego elementu tablicy
```

Adresowanie pamięci pośrednie:

```
my_table DB  10 dup(0)     ;alokacja w pamięci 10 bajtów, każdy inicjalizowany jako 0
mov EBX, offset my_table   ;adress my_table przeniesiony do reestru RBX
mov [EBX], 110             ;my_table[0] = 110
add EBX,   2               ;EBX = EBX +2
mov [EBX], 123             ;my_table[1] = 123
```