**Міністерство освіти і науки України Національний технічний університет України «Київський Політехнічний Інститут імені Ігоря Сікорського» Кафедра конструювання електронно-обчислювальної апаратури**

Звіт з виконання лабораторної роботи №3

з дисципліни “Основи мікропроцесорної техніки”

Виконав: студент групи ДК-82

Рудюк Б. Б.

Перевірила: ст.вик.

Бондаренко Н. О.

**Київ – 2020**

**Тема.** Знайомство з функціями вводу-виводу. Методи адресації даних. Обробка таблиць і рядків (ланцюжків) даних.

**Мета:** вивчити основні функції консольного вводу та виводу і навчитись використовувати їх в прикладних програмах мовою асемблера; освоїти методи адресації даних, написати програму яка зчитує дані (декілька слів) від користувача з консолі і виводить на екран тільки ті слова, в яких більше голосних ніж приголосних букв.

**Теоретична частина**

1. **Способи адресації.**
   1. **Реєстрова адресація**

Операнд (байт або слово) знаходиться в регістрі. Спосіб застосовується до всіх програмно - адресуємих регістрів процесора.

Приклад:

inc CH ;Плюс 1 до вмісту СН

push DS ;DS зберігається в стеку

mov ES, AX ;вміст АХ пересилається в ES

* 1. **Безпосередня адресація**

Операнд (байт або слово) вказується в команді і після трансляції надходить в код команди; він може мати будь-яке значення(число, адреса, код ASCII), а також бути представлений у вигляді символічного позначення.

Приклад:

mov АН, 40h; Число 40h завантажується в АН

mov AL, '\*'; Код ASCII символу '\*' завантажується в AL

int 21h; Команда переривання з аргументом 21h

limit = 528; Число 528 отримує позначення limit

* 1. **Пряма адресація пам'яті**

Адресується пам'ять: адреса комірки пам'яті (слова або байта) вказується в команді (зазвичай в символічній формі) і надходить в код команди. Приклад:

; Сегмент даних

mem1 dw 0; Слово пам'яті містить 0

mem2 db 230; Байт пам'яті містить 230

; Сегмент команд

inc mem1; Вміст слова mem1 збільшується на 1

mov DX, mem1; Вміст слова з ім'ям mem1 завантажується в DX

mov AL, mem2; Вміст байта з ім'ям mem2 завантажується в АL

Якщо потрібно звернутися до комірки пам'яті з відомою абсолютною адресою, то цю адресу можна безпосередньо вказати в якості операнда. Попередньо необхідно налаштувати будь-який сегментний регістр на початок тієї ділянки пам'яті, в якому знаходиться шукана комірка.

Приклад:

mov AX, 0; Настроїмо сегментний регістр ES

mov ES, AX; самий початок пам'яті (адреса 0)

mov AX, ES: [0]; АХ = вміст слова з адресою 0000h:0000h

mov DX, ES: [2]; DХ = вміст слова з адресою 0000h:0002h

Зауважимо, що в цьому випадку сегментний регістр треба вказувати обов'язково. Всі інші способи адресації відносяться до групи непрямої адресації пам'яті.

* 1. **Базова і індексна адресація пам'яті**

Відносний адреса комірки пам'яті знаходиться в регістрі, позначення якого заключається в квадратні дужки. При використанні регістрів ВХ або ВР адресацію називають базовою, при використанні регістрів SI або DI – індексною. При адресації через регістри ВХ, SI або DI як сегментного регістра, мається на увазі DS; при адресації через ВР - регістр SS. Таким чином, непряма адресація через регістр ВР призначена для роботи зі стеком. Однак при необхідності можна явно вказати необхідний сегментний регістр.

Приклад:

mov AL, [ВХ]; Сегментна адреса передбачається в DS, зміщення в ВХ

mov DL, ES: [ВХ]; Сегментна адреса в ES, зміщення в ВХ

mov DX, [ВР]; Сегментна адреса в SS, зміщення в ВР

mov AL, [DI]; Сегментна адреса в DS, зміщення в DI

* 1. **Базова і індексна адресації пам'яті зі зміщенням**

Відносний адрес операнда визначається сумою вмісту регістра (ВХ, ВР, SI або DI) і зазначеного в команді числа, яке називають зміщенням.

Приклад:

mas db 1,2,5,3,7,9,8,3,4 ;Масив символів

mov BX, 2 ;ВХ = індекс елемента в масиві

mov DL, mas [BX] ;В DL заноситься третій елемент масиву

Такий же результат дасть така послідовність команд:

mov BX, offset mas ;ВХ = відносний адрес комірки mas

mov DL, 2 [BX]

* 1. **Базово-індексна адресація пам'яті**

Відносний адрес операнда визначається сумою вмісту базового і індексного регістрів. Допускається використання наступних пар:

[BX] [SI]

[BX] [DI]

[ВР] [SI]

[ВР] [DI]

Якщо в якості базового регістра виступає ВХ, то в якості сегментного мається на увазі DS (перші дві команди); при використанні в якості базового регістра ВР сегментним регістром за замовчуванням призначається SS (другі дві команди). При необхідності можна явно вказати необхідний сегментний регістр.

Приклад:

mov ВХ, [ВР] [SI]; В ВХ засилається слово з стека (сегментна адреса в SS),

; А зміщення обчислюється як сума вмісту ВР і SI

mov BX, ES: [BP] [SI] ;В ВХ засилається слово з сегмента, адреса якого

; Знаходиться в ES, а зміщення обчислюється як

; сума вмісту ВР і SI

mov ES: [BX + DI], АХ; В осередок пам'яті, сегментна адреса якої

; Зберігається в ES, а зсув дорівнює сумі

; вмісту ВХ і DI, пересилається вміст АХ

* 1. **Базово-індексна адресація пам'яті зі зміщенням**

Відносний адрес операнда визначається сумою трьох величин: вмістом базового і індексного регістрів, а також додаткового зміщення. Допускається використання тих же пар регістрів, що і в базово-індексному способі; так само чинних правил визначення сегментних регістрів.

Приклад:

mov mas [ВХ] [SI], 10 ;Число 10 пересилається в комірку пам'яті,

; сегментна адреса якої зберігається в DS, а

; Зсув дорівнює сумі вмісту ВХ і SI і зміщення комірки ;mas

mov AX, [BP + 2 + DI] ; В АХ пересилається з стека слово,

; Зміщення якого дорівнює сумі ВР, DI і "добавки" 2

1. **Прапор напряму**

Стан цього прапора впливає на те, як в процесі виконання команд обробки строкових примітивів змінюються значення регістрів ESI і EDI. Якщо прапор скинутий, вони збільшуються на розмір оброблюваного операнда (1, 2 або 4 байта), а якщо встановлено, то зменшуються.

|  |  |  |
| --- | --- | --- |
| Значення прапорця | Регістри ESI і EDI | Обробка даних |
| Скинутий | Збільшуються | Від молодших адресів до старших |
| Встановлений | Зменшуються | Від старших адресів до старших |

Значення прапора напрямку можна явно задати за допомогою команд CLD і STD:

CLD ;скидує флаг направлення

STD ;встановлення флагу направлення

**Завдання:** написати програму яка зчитує дані (декілька слів) від користувача з консолі і виводить на екран тільки ті слова, в яких більше голосних ніж приголосних букв.

**Виконання завдання**

В сегменті даних створюємо “вхідну строку”, масив wordarray на 25 елементів, в яку будуть записані дані (декілька англійських слів), “вихідну строку” масив outarray на 25 елементів, в яку буде виводитися результат (слова в яких голосних більше ніж приголосних). За допомогою функції ReadString, заносимо дані з консолі в wordarray.

Далі відбувається пошук кількості голосних і приголосних букв у введених словах. Якщо потрібно знайти кількість голосних букв у слові, то в спеціально створену функцію letmatch передається масив під назвою arrgol, в якому записані всі голосні букви англійської мови. Далі в letmatch відбувається сканування і порівняння (функція repne scasb), з масивом голосних букв, кожної букви введеного слова. В результаті, коли програма доходить до кінця слова, вона записує в регістр ebx кількість знайдених голосних букв у слові. Отримане значення зберігаємо в масив golcnt. Майже така ж сама інструкція для пошуку кількості приголосних букв у слові, в функцію letmatch передається масив під назвою arrprug, в якому записані всі приголосні букви англійської мови. Знову відбувається сканування і порівняння кожної букви введенного слова, з масивом приголосних букв. В кінці також результат записується в регістр ebx. Отримані значення кількості голосних і приголосних слова, порівнюємо за допомогою функції сmp.

Якщо голосних більше ніж приголосних, програма продовжується далі. Відбувається копіювання слова у вихідну строку (масив outarray) за допомогою функції rep movsb. Вона копіює певну кількість байт (кількість задано в регістрі ecx) у вихідну строку, з дрежела, яким в нашому випадку слугує вхідна строка, масив wordarray. Після того, як слово було скопійовано, програма перевіряє чи немає ПРОБЕЛ після слова. Якщо буде кінець строки, викликається функція WriteString, яка виведе на екран вихідну строку в якій записані тільки ті слова, в яких голосних букв більше ніж приголосних. Якщо це не останнє слово, то відбувається перекид на мітку find і заново відбувається весь цикл перевірок для наступного слова.

Якщо приголосних більше ніж голосних, програма перескочить до мітки IsEnd, відбудеться перевірка чи після слова наявний ПРОБЕЛ. Якщо він відсутній, значить це кінець строки, викликається функція WriteString, яка виведе на екран вихідну строку в якій записані тільки ті слова, в яких голосних букв більше ніж приголосних. Якщо це не останнє слово, то відбувається перекид на мітку find і заново відбувається весь цикл перевірок для наступного слова.

Вихідний код

INCLUDE Irvine32.inc ;підключення бібліотеки Ірвіна

.data

arrgol BYTE "aeiouyAEIOUY",0 ;масив голосних букв

arrprug BYTE "bcdfghjklmnpqrstvwxzBCDFGHJKLMNPQRSTVWXZ",0 ;масив приголосних букв

golcnt DWORD 0 ;змінна, що зберігає кількість голосних

lengarrw DWORD 0 ;тимчасова змінна для збереження кількості елементів в масиві голосих або приголосних

temp DWORD 0 ;тимчасова змінна

wordarray BYTE 25 DUP(0) ;рядок, в який записують слова (масив на 25 комірок)

outarray BYTE 25 DUP(0) ;рядок, в який виводиться результат (масив на 25 комірок)

arraysize = (SIZEOF wordarray) - 1 ;довжина масива, що вводиться в рядок

ofswarr DWORD 0 ;змінна, що зберігає вказівник на початок масива wordarray

ofsoarr DWORD 0 ;змінна, що зберігає вказівник на початок масива outarray

.code

main PROC

mov edx, OFFSET wordarray ;адреса початку рядка

mov ecx, arraysize ;максимальна кількість символів, які може ввести користувач

call ReadString ;функція за допомогою якої зчитується строка

mov eax, OFFSET wordarray

mov ofswarr, eax

mov eax, OFFSET outarray

mov ofsoarr, eax

find:

mov esi, ofswarr

mov edi, OFFSET arrgol

mov edx, LENGTHOF arrgol

mov lengarrw, edx

call letmatch ;створена функція, за допомогою якої, порівнюємо кожну букву введеного слова з масивом голосних або приголосних

;при співпаданні букв, збільшує значення регістра еbx на 1

;у даному випадку порівнюється з масивом голосних

mov golcnt, ebx

mov esi, ofswarr

mov edi, OFFSET arrprug

mov edx, LENGTHOF arrprug

mov lengarrw, edx

call letmatch ;створена функція, за допомогою якої, порівнюємо кожну букву введеного слова з масивом голосних або приголосних

;при співпаданні букв, збільшує значення регістра еbx на 1

;у даному випадку порівнюється з масивом приголосних

cmp golcnt, ebx ;порівнює значення кількості голосних букв в слові з неголосними

;if(голосних > приголосних) флаг нуля ZR = 0 і флаг переносу СY = 0

;if(голосних < приголосних) флаг переносу СY = 1, else флаг нуля ZR = 1

jng IsEnd ;if(ZR = 1 or CY = 1) перекидує до мітки IsEND

;else не відбувається перекиду, програма продовжується далі

CopyWord:

sub esi,ofswarr ;розмір слова

cld ;зкидання флага направлення

mov ecx, esi ;запис скільки байтів буде скопійовано

mov esi, ofswarr ;адреса джерела(початок слова)

mov edi, ofsoarr ;адреса одержувача

rep movsb ;копіюємо певну кількість байтів, кількість яких вказано в регістрі ecx

mov ofsoarr, edi ;зміщення для запису наступного слова

IsEnd:

mov ofswarr, esi ;зміщення на наступне слово

mov dl, ' '

cmp [esi], dl ;перевірка чи це "ПРОБЕЛ", якщо так ZR = 1

jz find ;якщо ZR = 1 перекидує на мітку find

mov edx, OFFSET outarray

call WriteString ;функція за допомогою якої рядок результату можна вивести на екран

exit

main ENDP

letmatch PROC

mov temp, edi

mov ebx,0

Start:

mov edx, lengarrw ;довжина рядка, що буде скануватися (масив голосних або приголосних букв)

mov ecx, edx

mov eax, [esi] ;запис значення букви введеного слова

mov edi, temp ;адреса рядка, що буде скануватися (масив голосних або приголосних букв)

cld ;зкидання флага направлення

repne scasb ;функція, яка сканує масив (голосинх або приголосних букв), поки не знайде символ записаний в масиві еах

;if(не знайдено збігів) регістр ecx обнуляється (ZR = 0), else в регістр edi записується адреса найденого символа + 1

jnz isEnd ;якщо ZR = 0 перекидує на мітку isEnd

inc ebx

isEnd:

inc esi ;перехід на наступну букву

mov dl, ' '

cmp [esi],dl ;перевірка чи кінець слова

jz Finish

mov dl,0

cmp [esi],dl ;перевірка чи кінець строки

jz Finish

jmp Start

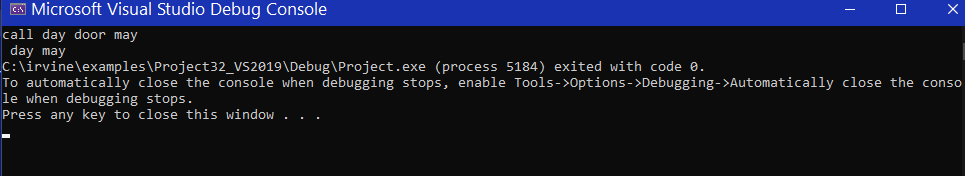
Finish:

ret ;після команди ret, процесор поверне управління наступній команді, яка знаходиться після тієї, що викликала процедуру letmatch

letmatch ENDP

END main

Вивід програми буде таким:



**Висновок**

Під час виконання цієї лабораторної роботи, я дізнався про методи адресації, та методи вводу-виводу за допомогою бібліотеки Irvine32.lib. Дізнався про операцію cmp та як вона влаштована, умовні переходи на мітки, ввод-вивід даних через консоль. Використовував регістрову адресацію, безпосередню адресацію, відносну-непряму адресацію та адресацію до стеку.