Учреждение образования
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ»
Кафедра информатики

Отчет по лабораторной работе №4

Работа со строками

Выполнил: студент гр. 853501
Астрашаб В.В.

Руководитель: ст. преподаватель
Шиманский В.В.

Минск 2019

СОДЕРЖАНИЕ

1. Введение

Целью данной работы является изучить следующий материал:

1) Представление строки в языке Ассемблер.

2) Команды movsb, movsw, stosb, stosw, lodsb, lodsw.

3) Назначение флага направления, команды CLD и STD.

4) Префиксы REP, REPE, REPNE.

2. Постановка задачи

2.1. Текст задания

С клавиатуры вводится строка, слова которой разделены пробелами. Необходимо перевернуть каждое ее слово задом наперед, а затем удалить все гласные буквы.

2.2. Условие задания

Необходимо ввести строку с клавиатуры, сделать ее обработку согласно заданию и показать результат на экране. При выполнении работы необходимо использовать хотя бы одну команду для работы с цепочками.

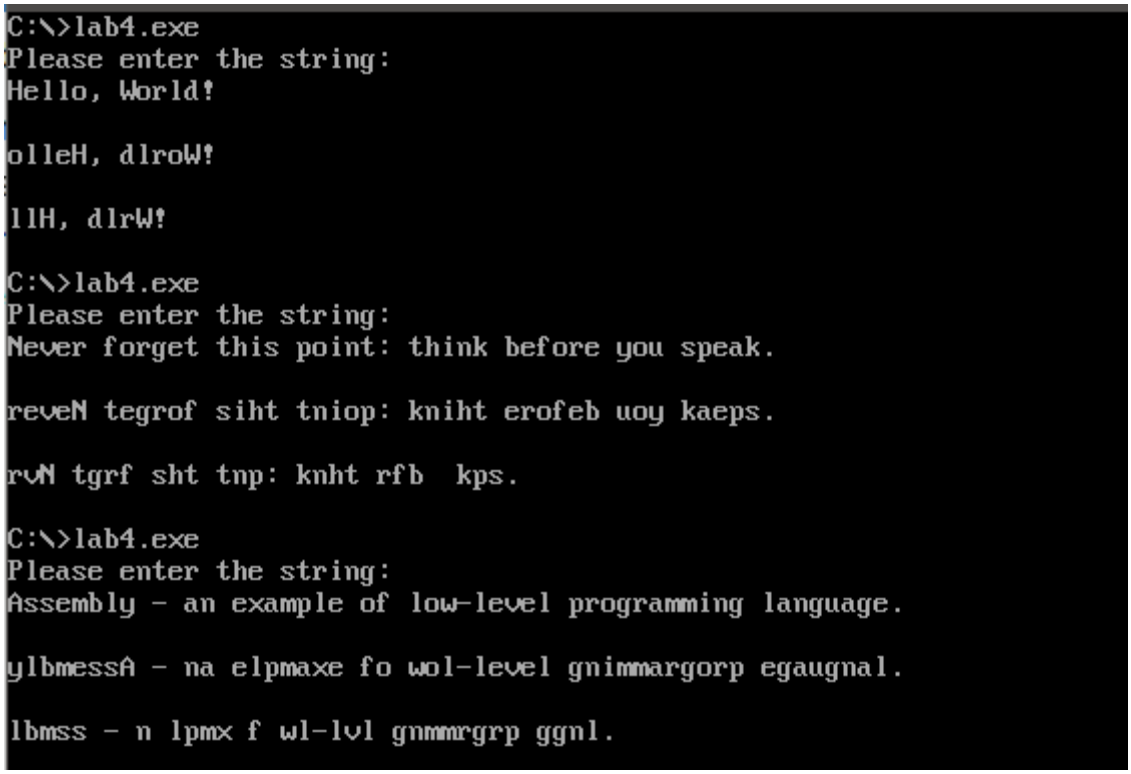
3. Программная реализация

3.1 Для реализации я использовал программу DOSBox на ОС Windows 10. DOSBox содержит компоненты TurboAssembler, TurboLink и TurboDebugger для компиляции, линковки и отладки программы.

3.2 Строка вводится пользователем с клавиатуры. Программа разбита на подпрограммы, каждая из которых выполняет определенную часть задания.

3.3 Ввод изначальной строки с клавиатуры происходит с помощью подпрограммы ReadString. Результат и промежуточная строка выводятся на экран при помощи подпрограммы PrintString.

3.4 Примеры:



Результат: Подпрограммы переворота слов и удаления гласных работают корректно. Программа сохраняет знаки препинания на их первоначальных местах.

4. Выводы

На практике я изучил и опробовал, в соответствии с поставленной задачей: представление строки в языке Ассемблер, команды movsb, movsw, stosb, stosw, lodsb, lodsw, назначение флага направления, команды CLD и STD, префиксы REP, REPE, REPNE.

В процессе выполнения лабораторной работы я использовал DOSBox и его компоненты TurboLink и TurboAssembler.

Для работы со строками я создал отдельные подпрограммы, которые используют команды работы с цепочками.

Для примеров я сделал скриншоты из программы DOSBox.

Литература

1. Юров В.И. – «Assembler. Учебник для вузов. 2-ое издание, 2003 год».
2. Юров В.И. – «Assembler. Практикум. 2-ое издание, 2006 год».
3. Калашников О.А. – «Ассемблер - это просто. 2-ое издание, 2011 год».

Приложение 1. Текст программы

```
model small
.stack 100h
.data
Prompt db 'Please enter the string: $'
Str1 db 100 dup('$')
Str2 db 100 dup('$')

.code

Interrupt proc
  mov ah, 4Ch
  int 21h
  ret
Interrupt endp


PrintPrompt proc
  lea dx, Prompt
  mov ah, 09h
  int 21h
  mov dl, 0Ah
  mov ah, 02h
  int 21h
  ret
PrintPrompt endp


PrintString proc
; Print a string from dx to console
  mov bx, dx
```

```
  mov dl, 0Ah
  mov ah, 02h
  int 21h

  mov dx, bx

  mov ah, 09h
  int 21h

  mov dl, 0Ah
  mov ah, 02h
  int 21h

  ret
PrintString endp


ReadString proc
; Read a string from console to di
CycleRead:
; Read a character
  mov ah, 01h
  int 21h

; Check for Enter
  cmp al, 0Dh
  jz EndCycle

; Save a character to the string
  stosb
```

```
  jmp CycleRead

EndCycle:
  mov al, '$'
  stosb
  ret
ReadString endp


IsVowel proc
; Determines if al is vowel. Returns the result in bx
  cmp al, 'a'
  jz ItIsVowel
  cmp al, 'e'
  jz ItIsVowel
  cmp al, 'i'
  jz ItIsVowel
  cmp al, 'o'
  jz ItIsVowel
  cmp al, 'u'
  jz ItIsVowel
  cmp al, 'y'
  jz ItIsVowel
  cmp al, 'A'
  jz ItIsVowel
  cmp al, 'E'
  jz ItIsVowel
  cmp al, 'I'
  jz ItIsVowel
  cmp al, 'O'
  jz ItIsVowel
```

```
  cmp al, 'U'
  jz ItIsVowel
  cmp al, 'Y'
  jz ItIsVowel

  mov bx, 0
  jmp EndVowel

ItIsVowel:
  mov bx, 1
EndVowel:
  ret
IsVowel endp



IsLetter proc
; Determines if al is a letter. Returns the result in
bx
  cmp al, 'a'
  jc CheckBigLetter
  cmp al, '{'
  jnc ItIsNotLetter
  jmp ItIsLetter

CheckBigLetter:
  cmp al, 'A'
  jc ItIsNotLetter
  cmp al, '['
  jnc ItIsNotLetter
  jmp ItIsLetter

ItIsLetter:
```

```
  mov bx, 1
  jmp EndLetter

ItIsNotLetter:
  mov bx, 0
EndLetter:
  ret
IsLetter endp


DeleteVowels proc
; Deletes vowels from si and saves the result to di
CycleDeleteVowels:
  lodsb
  cmp al, '$'
  jz StopDeleteVowels

  call IsVowel
  cmp bx, 0
  jnz CycleDeleteVowels

  stosb
  jmp CycleDeleteVowels

StopDeleteVowels:
  stosb
  ret
DeleteVowels endp


ReverseWords proc
```

```
; Reverses words from si and saves the result to di
  mov cx, 0
ReverseCycle:
  lodsb
  call IsLetter
  cmp bx, 1
  jnz NotLetterBranch
  push ax
  inc cx
  jmp ReverseCycle

NotLetterBranch:
  cmp cx, 0
  jz CXZero
  mov dx, ax
CacheLoop:
  pop ax
  stosb
  loop CacheLoop
  mov ax, dx
CXZero:
  stosb
  cmp al, '$'
  jz ReturnReverse
  jmp ReverseCycle

ReturnReverse:
  ret
ReverseWords endp
```

```
start:
  mov ax, @data
  mov ds, ax
  mov es, ax
  cld

  lea di, Str1
  call PrintPrompt
  call ReadString

  lea si, Str1
  lea di, Str2
  call ReverseWords

  lea dx, Str2
  call PrintString

  lea si, Str2
  lea di, Str1
  call DeleteVowels

  lea dx, Str1
  call PrintString

  call Interrupt
end start
```