|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | **Министерство науки и высшего образования Российской Федерации**  Калужский филиал  федерального государственного бюджетного  образовательного учреждения высшего образования  ***«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»***  ***(КФ МГТУ им. Н.Э. Баумана)*** |

**ФАКУЛЬТЕТ** ***ИУК «Информатика и управление»***

**КАФЕДРА** \_\_***ИУК4 «Программное обеспечение ЭВМ, информационные технологии»***

**ЛАБОРАТОРНАЯ РАБОТА №8**

**«Обработка строк»**

**ДИСЦИПЛИНА: «Машинно-зависимые языки программирования»**

|  |  |  |
| --- | --- | --- |
| Выполнил: студент гр. ИУК4-31Б | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (Отрошенко Т. В.)  (Подпись) |
| Проверил: | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (Амеличева К. А.)  (Подпись) |
| Дата сдачи (защиты):  Результаты сдачи (защиты): | | |
|  | - Балльная оценка:  - Оценка: | |

Калуга, 2021

**Цель работы:** практическое овладение навыками разработки программного кода на языке Ассемблер. Изучение организации обработки цепочек данных и команд строковых примитивов.

**Задача:** написать программу, которая реализует следующие функции:

1. Первая строка задана в сегменте данных (ST1), вторая символьная строка вводиться с клавиатуры (ST2), сравнить их, используя команду CMPS.

* Если строки совпадают, то выполнить задание а) индивидуального варианта (SCAS).
* Если не совпадают, выполнит задание б) индивидуального варианта (MOVS, STOS).

2. При выполнении каждого задания предусмотреть вывод на экран исходной строки и результатов преобразования.

3. Реализовать простейший интерфейс взаимодействия с пользователем, выполнять задание до выбора команды «Выход».

**Вариант 10**

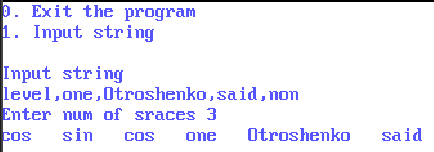
Слова в строке разделены запятыми. Вывести каждое слово на экран, разделив их между собой указанным числом пробелов.

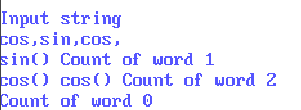
а) в строке ST2 найти слова «sin», «cos» или «log», вывести их на экран поставить после них символ “(“. Подсчитайте количество слов.

б) Переслать в строку ST1 слова из ST2, удалив все палиндромы (то есть при чтении наоборот содержание не изменяется, например, слово `БОБ').

|  |
| --- |
| .MODEL small  .STACK 100h  .486 ; Включает сборку инструкций для процессора 80386  mWriteStr macro string  push ax ; Сохранение регистров, используемых в макросе, в стек  push dx  mov ah, 09h ; 09h - функция вывода строки на экран  mov dx, offset string  int 21h  pop dx ; Перенос сохранённых значений обратно в регистры  pop ax  endm mWriteStr  mCLS macro start  push ax ; Сохранение регистров, используемых в макросе, в стек  push bx  push cx  push dx  mov ah, 10h  mov al, 3h  int 10h ; Включение режима видеоадаптора с 16-ю цветами  mov ax, 0600h ; ah = 06 - прокрутка вверх  mov bh, 11111001b ; белый фон, синий текст  mov cx, start ; ah = 00 - строка верхнего левого угла  mov dx, 184Fh ; dh = 18h - строка нижнего правого угла  int 10h ; Очистка экрана и установка цветов фона и текста  mov dx, 0 ; dh - строка, dl - столбец  mov bh, 0 ; Номер видео-страницы  mov ah, 02h ; 02h - функция установки позиции курсора  int 10h ; Устанавливаем курсор на позицию (0, 0)  pop dx ; Перенос сохранённых значений обратно в регистры  pop cx  pop bx  pop ax  endm mCLS  mWriteAX macro  local convert, write  push ax ; Сохранение регистров, используемых в макросе, в стек  push bx  push cx  push dx  push di  mov cx, 10 ; cx - основание системы счисления  xor di, di ; di - количество цифр в числе  or ax, ax ; Проверяем, равно ли число в ax нулю и устанавливаем флаги  jns convert ; Переход к конвертированию, если число в ax положительное  push ax  mov dx, '-'  mov ah, 02h ; 02h - функция вывода символа на экран  int 21h ; Вывод символа "-"  pop ax  neg ax ; Инвертируем отрицательное число  convert:  xor dx, dx  div cx ; После деления dl = остатку от деления ax на cx  add dl, '0' ; Перевод в символьный формат  inc di ; Увеличиваем количество цифр в числе на 1  push dx ; Складываем в стек  or ax, ax ; Проверяем, равно ли число в ax нулю и устанавливаем флаги  jnz convert ; Переход к конвертированию, если число в ax не равно нулю  write: ; Вывод значения из стека на экран  pop dx ; dl = очередной символ  mov ah, 02h  int 21h ; Вывод очередного символа  dec di ; Повторяем, пока di <> 0  jnz write  pop di ; Перенос сохранённых значений обратно в регистры  pop dx  pop cx  pop bx  pop ax  endm mWriteAX  mReadAX macro buffer, size  local input, startOfConvert, endOfConvert  push bx ; Сохранение регистров, используемых в макросе, в стек  push cx  push dx  input:  mov [buffer], size ; Задаём размер буфера  mov dx, offset [buffer]  mov ah, 0Ah ; 0Ah - функция чтения строки из консоли  int 21h  mov ah, 02h ; 02h - функция вывода символа на экран  mov dl, 0Ah  int 21h ; Переносим курсор на новою строку  xor ah, ah  cmp ah, [buffer][1] ; Проверка на пустую строку  jz input ; Если строка пустая - переходим обратно к вводу  xor cx, cx  mov cl, [buffer][1] ; Инициализируем переменную счетчика  xor ax, ax  xor bx, bx  xor dx, dx  mov bx, offset [buffer][2] ; bx = начало строки (строка начинается со второго байта)  cmp [buffer][2], '-' ; Проверяем, отрицательное ли число  jne startOfConvert ; Если отрицательное - пропускаем минус  inc bx  dec cl  startOfConvert:  mov dx, 10  mul dx ; Умножаем на 10 перед сложением с младшим разрядом  cmp ax, 8000h ; Если число выходит за границы, то  jae input ; возвращаемся на ввод числа  mov dl, [bx] ; Получаем следующий символ  sub dl, '0' ; Переводим его в числовой формат  add ax, dx ; Прибавляем к конечному результату  cmp ax, 8000h ; Если число выходит за границы, то  jae input ; возвращаемся на ввод числа  inc bx ; Переходим к следующему символу  loop startOfConvert  cmp [buffer][2], '-' ; Ещё раз проверяем знак  jne endOfConvert ; Если знак отрицательный, то  neg ax ; инвертируем число  endOfConvert:  pop dx ; Перенос сохранённых значений обратно в регистры  pop cx  pop bx  endm mReadAX  ; макросы относящиеся к данной работе  mEqual macro str1, str2; результат в ax или прямая передача управления подфункциям  local neq, exit\_  pusha  mov si, offset str1+2  mov di, offset str2+2  mov al, [si-1]  cmp al, [di-1]  jnz neq  xor cx, cx  mov cl, al  rep cmpsb  jnz neq  ; mov ax, 1  mFind str2, str\_sin  mFind str2, str\_cos  mFind str2, str\_log  jmp exit\_  neq:  ; mov ax, 0  mAddStrPal str2, str1, buffer  mOutputStr str1  mWriteStr endl  exit\_:  popa  endm mEqual  mReadStr macro string ; считывает строку с консоли. dw длина db строка  push ax  push bx  push dx  mWriteStr str\_input  xor ax, ax  mov dx, offset string  mov ah, 0Ah  int 21h    xor bx,bx  mov bl,string[1]  mov string[2+bx],'$'  mWriteStr endl  pop dx  pop bx  pop ax  endm mReadStr  mLenStr macro string ; считает длину строки вместе с завершающим символом и помещает в сх  push di  push ax  mov di, offset string  mov al, '$'  MOV CX, 1000h  repne SCASb  mov cx, offset string  sub di, cx  mov cx, di  push ax  push di  endm  mOutputStr macro string ; вывод пропускает длину строки  local next\_el, exit\_, print, space\_loop  push ax  push bx  push cx  push dx  push si  mWriteStr str\_space  mov ah, 01h  int 21h  sub al, '0'  mov ah, 0  mov bx, ax ; количество пробелов  mWriteStr endl  mov si, offset string  add si, 2 ; пропуск длины строки (первые два байта)  next\_el:  lodsb ; загрузка буквы из строки  cmp al, '$'  jz exit\_  cmp al, ','  jnz print  ; вывод пробелов в случае ","  mov cx, bx  space\_loop:  mWriteStr space  loop space\_loop  jmp next\_el  print: ; вывод буквы  mov ah, 02h  mov dl, al  int 21h  jmp next\_el  exit\_:  mWriteStr endl  pop si  pop dx  pop cx  pop bx  pop ax  endm mOutputStr  mFind macro string, find\_string  local next\_word, exit\_2  pusha  lea si, string  lodsw  mov cx, ax ; длину обычной строки  xchg cl, ch  xor bx, bx ; счетчик искомых слов  mov al, ','  mov di, si  next\_word:  mov si, di ; si хранит начало слова  repne scasb ; находит начало следующего слова  ; ( соответвенно конец предыдущего на 2 меньше si)  jne exit\_2 ; провека достигнут ли конец строки (cx = 0)  push cx  mov cx, di  sub cx, si ; длина слова  dec cx  push di  push si  lea di, find\_string  repe cmpsb  pop si  pop di  pop cx  jne next\_word  mWriteStr find\_string  mWriteStr brase  inc bx  jmp next\_word  exit\_2:  mov ax, bx  mWriteStr str\_count  mWriteAX  mWriteStr endl  popa  endm mFind  mPalindrome macro string  local main\_loop, end\_main\_loop, сheck\_palindrome, сheck\_palindrome\_loop, delete\_loop, end\_delete\_loop, new\_word, end\_macro  pusha  lea si, string  lodsw  mov cx, ax  mov dx, si  xor cx, cx  main\_loop:  cld  lodsb  cmp al, '$'  je сheck\_palindrome  cmp al, 0Dh  je сheck\_palindrome  cmp al, ','  je сheck\_palindrome  inc cx  jmp main\_loop  сheck\_palindrome:  push si  mov di, si  sub di, 2  sub si, cx  dec si  mov ax, cx  mov cl, 2  div cl  mov cl, al  xor ch, ch  mov bl, al  xor bh, bh  сheck\_palindrome\_loop:  cld  lodsb  cmp al, byte ptr [di]  jne new\_word  dec di  loop сheck\_palindrome\_loop  sub si, bx  add di, bx  inc di  xchg si, di  cmp byte ptr [si], '$'  je end\_main\_loop  cmp byte ptr [si], 0Dh  je end\_main\_loop  inc si  delete\_loop:  movsb  cmp byte ptr [si], '$'  je end\_delete\_loop  cmp byte ptr [si], 0Dh  je end\_delete\_loop  jmp delete\_loop  end\_delete\_loop:  movsb  pop si  mov si, dx  xor cx, cx  jmp main\_loop  new\_word:  pop si  mov dx, si  xor cx, cx  dec si  cmp byte ptr [si], '$'  je end\_macro  cmp byte ptr [si], 0Dh  je end\_macro  inc si  jmp main\_loop  end\_main\_loop:  movsb  end\_macro:  mLenStr string  lea si, string+1  sub cl, 2  mov [si], cl  popa  endm  mAddStrPal macro sour, dest, buff  pusha  xor cx,cx  lea si, sour  lea di, buff  mov cl, [si+1]  add cl, 2  rep movsb  mov ax, 242Ch  stosw  mPalindrome buff  lea si, dest+1  lodsb  add si, ax  mov di, si  lea si, buffer+1  lodsb  mov cl, al  rep movsb  popa  endm mAddStrPal  .DATA  buffer db 50 dup('0')  endl db 13, 10, '$'  tab db 09, '$'  space db ' $'  brase db '() $'  str\_space db 'Enter num of sraces $'  menuInstruction db '0. Exit the program', 13, 10, '1. '  str\_input db 'Input string', 13, 10, '$'  str\_count db 'Count of word $'  str1 db 0, 12, "cos,sin,cos,$", 20 dup ('0')  str2 db 50 dup ('0')  str\_sin db "sin$"  str\_cos db "cos$"  str\_log db "log$"  .CODE  Start:  mov ax, @data  mov ds, ax  mov es, ax  mCLS 0000b ; Макрос очистки экрана и установки вида окна  mWriteStr menuInstruction ; Макрос вывода строки на экран  mWriteStr endl  menu: ; Вывод на экран меню, а также осуществление выбора следующего пункта программы  mov ah, 00h  int 16h ; Ожидание нажатия символа и получение его значения в al  cmp al, "0"  je exit  jmp consoleInput  jmp menu  consoleInput: ; Ввод элементов массива из консоли  mReadStr str2  mEqual str1, str2  jmp menu  exit: ; Завершение программы  mov ax, 4c00h  int 21h  end Start |

**Результаты выполнения:**

****

****

**Вывод:** в ходе выполнения лабораторной работы были приобретены навыки написания программ с циклами на языке Ассемблер и изучена организация обработки цепочек данных и команд строковых примитивов.