НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»
Факультет компьютерных наук, Департамент программной инженерии
Курс: «Архитектура вычислительных систем»

Программирование на языке ассемблера

Микропроект

Коновалов Егор Андреевич,
Студент БПИ195

1. ТЕКСТ ЗАДАНИЯ

Разработать программу, определяющую список одинаковых символов, содержащихся в двух заданных ASCII строках.

2. ПРИМЕНЁННЫЕ РАСЧЕТНЫЕ МЕТОДЫ

Шаги алгоритма для нахождения списка одинаковых символов в строках:

1. Создать массив been1[256] и been2[256] для определения символов встречающихся в первой и второй строке соответственно.
2. Заполнить массивы, пройдя по строкам так, что для символа s, принадлежащего строке been[s] = <количество появлений этого символа в троке>.
3. Пройтись по массивам been1[] и been2[] и вывести каждый символ, для которого соответствующее значение в обоих массивах больше нуля.

3. ИСПОЛЬЗОВАННЫЕ ИСТОЧНИКИ

1. Руководство программиста Flat Assembler: https://flatassembler.net/docs.php?article=manual
2. Материал курса «Архитектура вычислительных систем»: http://www.softcraft.ru/edu/comparch/practice/asm86/06-str
3. Сайт, посвященный программированию на языках низкого уровня: http://av-assembler.ru/

4. ТЕКСТ ПРОГРАММЫ

```
format PE console
entry start

include 'win32a.inc'

; ==========================================================================

section '.data' data readable writable
        str1         rb   255           ; reserved memory for input string
        str2         rb   255           ; same as above
        scanStr      db   '%s', 0       ; scanf format
        yes          db   'yes', 10, 0
        no           db   'no', 10, 0
        charformat   db   '%c', 0       ; printf format for single char
        endchar      db   10, 0         ; end of line char
        tmp          dd   4             ; variable for temporary data
        i            dd   4             ; variable for storing temporary ecx

        been1        rb   256           ; array been[<ascii-code>] = <if (char was
processed) 1 else 0>
        been2        rb   256           ; array been[<ascii-code>] = <if (char was
processed) 1 else 0>
        beenSize     dd   256           ; size of the array been
```

```
        enterString1 db   'Enter first string (max length 100): ', 0
        enterString2 db   'Enter second string (max length 100): ', 0
        pressAnyKey  db   10, 'Press any key to quit.', 0
        commonCharsS db   10, 'Common chracters of two strings in ASCII table
order:', 10, 0
        errorStringSize   db  'String length must be lower than or equal to 255.',
10, 0

; ========================================================================
section '.code' code readable executable
start:
        invoke printf, enterString1
        add esp, 4
        invoke scanf, scanStr, str1     ; read first char sequence
        add esp, 8

        stdcall strlen, str1
        add esp, 4
        cmp eax, 255                    ; check if string size is in range
        jg incorrectStringSize

        invoke printf, enterString2
        add esp, 4
        invoke scanf, scanStr, str2     ; read second char sequence
        add esp, 8

        stdcall strlen, str2
        add esp, 4
        cmp eax, 255                    ; check if string size is in range
        jg incorrectStringSize

        stdcall markCharsContained, str1, been1 ; mark characters containde in the
first string
        add esp, 8

        stdcall markCharsContained, str2, been2 ; mark characters containde in the
second string
        add esp, 8

        invoke printf, commonCharsS
        add esp, 4

        call printCommon                ; print common characters

        invoke printf, pressAnyKey      ; wait for user to press any key before exit
        add esp, 4
exit:
        invoke getch
        invoke ExitProcess, 0
incorrectStringSize:
        invoke printf, errorStringSize
        jmp exit
; ------------------------------------------------------------------------
; markCharsContained(string, byte[])
; Procedure that marks chars contained in string.
; if char in string:
;    arr[char] = 1
```

```
markCharsContained:
        mov esi, [esp + 4]              ; get string
        mov ebx, [esp + 8]              ; get array
markCharsContainedLoop:
        xor al, al                      ; clear al register
        cmp al, [esi]                   ; check if [esi] == 0 == end of line
        je markCharsContainedExit       ; if equals jump to exit

        stdcall markBeenInArray, [esi], ebx    ; mark character as been
        add esp, 8

        inc esi                         ; move to the next char
        jmp markCharsContainedLoop
markCharsContainedExit:
        ret
; ------------------------------------------------------------------------
; markBeenInArray(char, byte[])
; Procedure that increments byte[char]
markBeenInArray:
        mov al, [esp+4]                 ; eax = char argument
        mov ebx, [esp+8]                ; ebx = ascii array
        mov [tmp], ebx                  ; temporary save arr pointer to restore it
        add ebx, eax                    ; move pointer to position of the character
        mov al, byte [ebx]              ; get current number of character appearances
        inc al                          ; inc been[<ascii char>]
        mov [ebx], al                   ; save new value of been[<ascii char>]
        mov ebx, [tmp]                  ; restore previous value of ebx
        ret
; ------------------------------------------------------------------------
; printCommon()
; Procedure that prints common characters of two string
; by iterating through been arrays and checking if both characters
; are marked, i. e.
; for i in 0..(beenSize = 256):
;    if (been1[i] > 0 && been2[i] > 0):
;       print("%c", i)
printCommon:
        xor ecx, ecx                    ; ecx = 0
        mov eax, been1                  ; eax points to been array of the first
string
        mov ebx, been2                  ; ebx points to been array of the second
string
printCommonLoop:
        mov [tmp], ebx                  ; save value of ebx to use bl for comparison
        xor ebx, ebx
        mov bl, byte [eax]
        cmp bl, 0                       ; check if char = ecx is marked in the array
been1
        jg printCommonPresentInBeen     ; if so, jump to next step
        mov ebx, [tmp]
printCommonAfterIf:
        inc ecx                         ; increment pointers and counter
        inc eax
        inc ebx
        cmp ecx, [beenSize]             ; if counter if greater than array size -
exit
        jne printCommonLoop             ; beenSize = been1.size = been2.size = 256
```

```
        ret
printCommonPresentInBeen:
        mov ebx, [tmp]                  ; restore previously saved value of ebx
        mov [tmp], eax                  ; save value of ebx to use al for comparison
        mov al, byte [ebx]
        cmp al, 0                       ; check if char = ecx is marked in the array
been2
        jg printCommonAreCommon         ; if so, character is common for two string
        mov eax, [tmp]                  ; restore previously saved value of eax
        jmp printCommonAfterIf
printCommonAreCommon:
        mov [i], ecx                    ; save counter because it is used in printf
procedure
        invoke printf, charformat, ecx  ; print common character
        add esp, 8
        mov ecx, [i]                    ; restore counter value
        mov eax, [tmp]                  ; restore previously saved value of eax
        jmp printCommonAfterIf          ; return to the loop
;-------------------------------------------------------------------------
; strlen(string)
; Strlen procedure (as strlen in C programming language library)
strlen:
    mov     edi, [esp+4]                ; using msg as stack argument
    mov     ecx, -1                     ; ecx must be < 0
    xor     al, al                      ; tail symbol is zero
    cld                                 ; direction from begin to end
    repne   scasb                       ; while(msg[edi] != al) {edi++; ecx--;}
    neg     ecx
    sub     ecx, 2                      ; ecx = length(msg)-2
    mov     eax, ecx
    ret

; ========================================================================

section '.idata' import data readable
    library kernel, 'kernel32.dll',\
            msvcrt, 'msvcrt.dll',\
            user32,'USER32.DLL'

include 'api\user32.inc'
include 'api\kernel32.inc'
    import kernel,\
           ExitProcess, 'ExitProcess'
  include 'api\kernel32.inc'
    import msvcrt,\
           printf, 'printf',\
           scanf, 'scanf',\
           getch, '_getch'
```