Федеральное государственное автономное

образовательное учреждение

Высшего образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

|  |
| --- |
| Институт космических и информационных технологий |
| институт |
| Программная инженерия |
| кафедра |

**ОТЧЕТ О ПРАКТИЧЕСКОЙ РАБОТЕ**

|  |
| --- |
| Строки и массивы |
| тема |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Преподаватель | |  |  |  | М. В. Сарамуд |
|  | |  | подпись, дата |  | инициалы, фамилия |
| Студент | КИ21-17/1Б, 032156940 |  |  |  | Н. А. Самарин |
|  | номер группы, зачётной книжки |  | подпись, дата |  | инициалы, фамилия |

Красноярск 2024

**СОДЕРЖАНИЕ**

1 Задание............................................................................................................... 3

2 Вариант.............................................................................................................. 3

3 Ход работы........................................................................................................ 3

4 Вывод................................................................................................................. 8

**1 Задание**

Разработать программы на Ассемблере, реализующие задания по  
массивам и строкам. Допускается реализация как в одной, так и в двух  
независимых программах. Выполните упражнения из ниже приведенного  
списка, выбирая вариант соответственно номеру студента в группе. Для  
задания со строками – под строкой подразумеваем последовательность  
символов, которые образуют слова и разделяющего знака – пробела. Слово -  
последовательность символов, ограниченная знаком пробела.

**2 Вариант**

Вариант 24. Среди строк целочисленной матрицы, которые содержат  
только простые числа, найти строку с максимальной суммой элементов.

Пользователь вводит строку, состоящую из слов, разделенных символом  
пробела, найти самое длинное слово, вывести его на экран, если слов  
максимальной длины несколько – любое.

**3 Ход работы**

В работе был использован FASM на операционной системе Windows 10.  
Ниже представлен код разработанных программ и скриншоты их работы.

Листинг 1 – Код программы для первого задания

format PE Console  
entry start  
include 'win32a.inc'  
section '.data' data readable writeable  
 rows dd 0  
 columns dd 0  
 row dd 0  
 column dd 0  
 maxSize dd 100  
 maxSum dd 0  
 sum dd 0  
 maxIndex dd 0  
 curIndex dd 0  
 matrix rb 400  
 inputFormat db "%5d", 0  
 outputFormat db "%d ", 0  
 newline db 10, 0   
 errorMessage db "Input error", 0  
 primeErrorMessage db "Element is not Prime error", 0  
 matrixMessage db "Matrix:", 10, 0  
 promptRows db "Enter number of rows: ", 0  
 promptColumns db "Enter number of columns: ", 0  
 promptElement db "Enter element at row %d, column %d: ", 0  
 maxSumMessage db "Maximum sum of elements: %d", 10, 0  
 maxRowMessage db "Row with the maximum sum of elements: ", 0  
section '.code' code readable writeable executable  
start:  
 call inputMatrix  
inputMatrix:  
 invoke printf, promptRows  
 invoke scanf, inputFormat, rows

Продолжение листинга 1

cmp [rows], 0  
 jbe error  
 invoke printf, promptColumns  
 invoke scanf, inputFormat, columns   
 cmp [columns], 0  
 jbe error  
 mov eax, [rows]  
 imul eax, [columns]  
 cmp eax, [maxSize]  
 jg error  
 mov esi, matrix  
 mov [row], 1  
 jmp outerLoop  
outerLoop:  
 mov [column], 1  
 jmp innerLoop  
innerLoop:  
 invoke printf, promptElement, [row], [column]  
 invoke scanf, inputFormat, esi  
 jmp checkPrime  
checkPrime:  
 cmp dword [esi], 1  
 jbe primeError  
 cmp dword [esi], 2  
 je continueInput  
 mov ebx, 2  
 jmp divideLoop  
divideLoop:  
 xor edx, edx  
 mov eax, dword [esi]  
 idiv ebx   
 cmp edx, 0   
 je primeError  
 inc ebx   
 cmp ebx, dword [esi]   
 jb divideLoop  
 jmp continueInput  
continueInput:  
 add esi, 4   
 inc [column]  
 mov eax, [column]  
 cmp eax, [columns]  
 jbe innerLoop  
 inc [row]  
 mov eax, [row]  
 cmp eax, [rows]  
 jbe outerLoop  
 mov esi, matrix  
 mov [row], 1  
 invoke printf, matrixMessage  
 jmp printLoop   
printLoop:  
 mov [column], 1  
 jmp printLoopInner  
printLoopInner:  
 invoke printf, outputFormat, dword [esi]  
 add esi, 4  
 inc [column]

Продолжение листинга 1

mov eax, [column]  
 cmp eax, [columns]  
 jbe printLoopInner  
 invoke printf, newline  
 inc [row]  
 mov eax, [row]  
 cmp eax, [rows]  
 jbe printLoop  
 mov esi, matrix  
 mov [row], 1  
 jmp sumLoop  
sumLoop:  
 mov [column], 1  
 mov [curIndex], esi  
 mov [sum], 0  
 jmp sumLoopInner  
sumLoopInner:  
 mov eax, [sum]  
 add eax, dword [esi]  
 mov [sum], eax  
 add esi, 4  
 inc [column]  
 mov eax, [column]  
 cmp eax, [columns]  
 jbe sumLoopInner  
 mov eax, [sum]  
 cmp eax, [maxSum]  
 jg updateSum  
 jmp incRow  
updateSum:  
 mov eax, [sum]  
 mov [maxSum], eax  
 mov eax, [curIndex]  
 mov [maxIndex], eax  
 jmp incRow  
incRow:  
 inc [row]  
 mov eax, [row]  
 cmp eax, [rows]  
 jbe sumLoop  
 invoke printf, maxRowMessage  
 mov [column], 1  
 mov esi, [maxIndex]   
 jmp printResult  
printResult:  
 invoke printf, outputFormat, dword [esi]  
 add esi, 4  
 inc [column]  
 mov eax, [column]  
 cmp eax, [columns]  
 jbe printResult  
 invoke printf, newline  
 invoke printf, maxSumMessage, [maxSum]  
 jmp exit   
error:  
 invoke printf, errorMessage  
 jmp exit  
primeError:

Окончание листинга 1

invoke printf, primeErrorMessage  
 jmp exit  
exit:  
 invoke getch  
 invoke ExitProcess, 0  
section '.idata' data import readable  
 library kernel, 'kernel32.dll',\  
 msvcrt, 'msvcrt.dll'  
import kernel,\  
 ExitProcess, 'ExitProcess'  
import msvcrt,\  
 printf, 'printf',\  
 getch, '\_getch',\  
 scanf, 'scanf',\  
 atoi, 'atoi',\  
 strcat, 'strcat',\  
 sprintf, 'sprintf'

Листинг 2 – Код программы для второго задания

format PE console 4.0  
entry start  
include 'win32a.inc'  
define stringBuf 512  
section '.data' data readable writable  
 charPrint db '%c', 0  
 messegeString db 'Enter string: ', 10, 0  
 inputString db stringBuf dup(0)  
 longWordOffset dd 0  
 stringLen dd 0  
section '.code' code readable executable  
findLongWord:  
 cld ; DF (флаг направления) = 0  
 mov ax, ds  
 mov es, ax  
 lea edi, [inputString]  
 mov al, 0  
 mov ecx, stringBuf  
 repnz scasb  
 ; высчитываем длину строки  
 mov eax, stringBuf  
 sub eax, ecx  
 mov [stringLen], eax   
 mov ecx, eax  
 mov ebx, ecx ; текущее начало слова (относительно счётчика ECX)  
 mov edx, 0 ; длина наибольшего слова  
 lea edi, [inputString]  
 findWord:  
 mov al, 32 ; пробел  
 repnz scas  
 ; сравнить длину слова  
 mov eax, ebx  
 sub eax, ecx  
 dec eax ; убираем учтенный пробел   
 cmp eax, edx ; в флаг ZF  
 jle notSetNewLen ; переход если SF != OF или ZF = 1 (a <= b), если текущ  
 ее слово не больше   
 ; если больше  
 mov edx, eax ; новая максимальная длина слова

Окончание листинга 2

; вычисляем величину смещения относительно начала   
 mov eax, [stringLen]  
 sub eax, ebx  
 mov [longWordOffset], eax  
 notSetNewLen:  
 mov ebx, ecx   
 cmp ecx, 0  
 jne findWord  
 ret   
printLongWord:  
 cld  
 mov eax, [longWordOffset]  
 mov eax, [longWordOffset]  
 lea esi, [inputString+eax]  
 ; длина до конца строки  
 mov eax, [stringLen]  
 sub eax, [longWordOffset]  
 mov ecx, eax  
 dec ecx ; не учитывая конец строки   
 printEachSymbol:  
 lodsb ; загрузить текущий символ в al  
 cmp al, 32  
 je printEnd ; если пробел - конец слова  
 push ecx  
 cinvoke printf, charPrint, eax  
 pop ecx  
 loop printEachSymbol   
 printEnd:  
 mov eax, 10  
 cinvoke printf, charPrint, eax   
 ret  
start:  
 cinvoke printf, messegeString  
 cinvoke gets, inputString  
 call findLongWord  
 call printLongWord  
 invoke getch  
 invoke ExitProcess, 0  
section '.idata' import data readable  
 library msvcrt,'MSVCRT.DLL',\  
 kernel32,'KERNEL32.DLL'  
 import kernel32,\  
 ExitProcess, 'ExitProcess',\  
 sleep,'Sleep'  
 import msvcrt,\  
 gets,'gets',\  
 printf,'printf', \  
 getch, '\_getch'

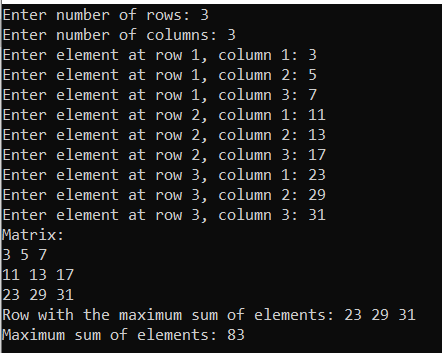


Рисунок 1 – Работа программы 1

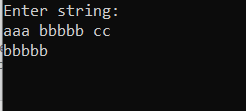


Рисунок 2 – Работа программы 2

**4 Вывод**

После изучения необходимой теории задание было выполнено.