**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра информационной безопасности**

отчет

**по учебной практике**

Тема: Основы исследования функциональности программного обеспечения и разработки программного обеспечения на ассемблере

|  |  |  |
| --- | --- | --- |
| Студент гр. 8363 |  | Нерсисян А.С. |
| Руководитель |  | Халиуллин Р.А. |

Санкт-Петербург

2020

**ЗАДАНИЕ**

**на Учебную практику**

|  |  |  |
| --- | --- | --- |
| Студент Нерсисян А.С. | | |
| Группа 8363 | | |
| Тема практики: Основы исследования функциональности программного обеспечения и разработки программного обеспечения на ассемблере | | |
| Задание на практику:  Изучить основные инструкции процессора архитектуры IA-32 (x86) и реализовать приложение на языке ассемблера. Объем и функциональность приложения для реализации — по выбору студента. Компилятор ассемблера и операционная система, для которой будет реализовано приложение — по выбору студента. | | |
| Сроки прохождения практики: 29.06.2020 — 12.07.2020 | | |
| Дата сдачи отчета: 17.07.2020 | | |
| Дата защиты отчета: 17.07.2020 | | |
|  | | |
| Студент |  | Нерсисян А.С. |
| Руководитель |  | Халиуллин Р.А. |

**Аннотация**

Целью данной практики является изучение основных инструкций процессора архитектуры IA-32 (x86) и реализация приложения на языке ассемблера. В рамках данной практики было разработано консольное приложение для работы с матрицами. В программе реализованы инструкции, которые позволяют читать, хранить, сортировать, а также вывести на экран матрицу, размеры которого не превышают 5х5. Отчет содержит в себе исходный код программы на языке ассемблера (Microsoft Macro Assembler).

**Summary**

The purpose of this practice is to study the basic instructions of the processor architecture IA-32 (x86) and implement the application in assembly language. As part of this practice was developed a console application for working with matrices. The program implements instructions to read, store, sort, and also display a matrix whose dimensions do not exceed 5 by 5. This report contains the source code of the program in the assembly language (Microsoft Macro Assembler).

**содержание**

|  |  |  |
| --- | --- | --- |
|  | Введение | 5 |
| 1. | Описание реализованного приложения | 6 |
| 1.1. | Описание функциональности реализованного приложения | 6 |
| 1.2. | Описание инструкций процессора, использованных в реализованном приложении | 6 |
| 2. | Результаты тестирования реализованного приложения | 13 |
|  | Заключение | 14 |
|  | Список использованных источников | 15 |
|  | Приложение 1. Исходный код реализованного приложения на ассемблере | 16 |

**введение**

Целью данной практики является изучение основных инструкций процессора архитектуры IA-32 (x86) и реализация приложения на языке ассемблера.

В рамках данной практики было разработано консольное приложение для работы с матрицами. В программе реализованы инструкции, которые позволяют читать, хранить, сортировать, а также вывести на экран матрицу, размеры которого не превышают 5х5.

**1. Описание реализованного приложения**

**1.1. Описание функциональности реализованного приложения**

Консольное приложение для работы с матрицами. При запуске программы на экран выводится сообщение, где от пользователя требуется ввести количество строк и столбцов матрицы (макс. 5х5), затем нужно ввести и саму матрицу.

После получения матрицы программа выводит на экран изначально введенную пользователем и отсортированную по возрастанию элементов матрицы.

Для разработки данного приложения в качестве текстового редактора было выбрано Microsoft Visual Studio Code.

Компилятор: Microsoft Macro Assembler 6.14.8444 (MASM).

Язык программирования: Assembler (MASM).

Операционная система: Microsoft Windows 10.

**1.2. Описание инструкций процессора, использованных в реализованном приложении**

При запуске программы в памяти резервируется место для данных с помощью следующих инструкций

mov ax, @data

mov ds,ax

Потом вызываются инструкции для ввода размеров входной матрицы, call input\_width и call input\_height для столбцов и строк соответсвенно.

Блок инструкций input\_width содержит в себе следующее: изначально на экран выводится текст (сообщение) о программе и ожидается ввод количества столбцов матрицы

This program created by Arthur Nersisyan for educational practice.

Enter matrix WIDTH(max 5).

input\_width proc

mov ah,09h ;09h вывод строки на экран

lea dx,msg1 ;перемещение строки в регистр

int 21h ;вызов прерывания

После нажатия любой клавиши на клавиатуре, скан код данной кнопки попадает в регистр al, поэтому значение нажатой кнопки можно оттуда получить, сравнивать с интересующими нам значениями и сделать условный переход.

iw\_1: mov ah,1

int 21h

cmp al,'1'

jl iw\_1

cmp al,'5'

jg iw\_1 ;JL, JG for cmp signed int,

sub al,'0'

sub ah,ah

mov cx,ax

ret

input\_width endp

В результате в регистр cx попадает значение (число, количество столбцов матрицы (от 1 до 5)).

Аналогичным образом в регистр bx попадает число строк матрицы.

input\_height proc

        mov ah,09h

        lea dx,msg2

        int 21h

ih\_1:   mov ah,1

        int 21h

        cmp al,'1'

        jl ih\_1

        cmp al,'5'

        jg ih\_1

        sub al,'0'

        sub ah,ah

        mov bx,ax

        ret

input\_height endp

Для начала ввода значений элементов матрицы нужно поместить в стек значения количества строк и столбцов матрицы:

push bx ;помещаем данные о размерах матрицы в стек

push cx

lea di,matrix1

call input\_matrix ;вызываем блок инструкций для ввода элементов

di – данные

cx – количество столбцов

bx – количество строк

С помощью ниже представленных инструкций постепенно в программу вводится матрица, после ввода каждого элемента на экран выводится сообщение «Enter number N». После ввода всей матрицы сообщение «Enter number N» больше не будет выводится на экран.

input\_matrix proc

mov si,10

sub ax,ax

mov dl,cl

mov dh,bl

im\_0:

push cx

push di

im\_1:

push dx

push di

lea di,msg5\_w

mov al,dl

sub al,cl

add al,'1'

mov [di],al

lea di,msg5\_h

mov al,dh

sub al,bl

add al,'1'

mov [di],al

pop di

mov ah,09h

lea dx,msg5

int 21h

pop dx

sub bp,bp

im\_2:

im\_3: mov ah,1

int 21h

cmp al,0Dh

je im\_5

cmp al,'0'

jl im\_3

cmp al,'9'

jg im\_3

push ax

push dx

mov ax,bp

mul si

mov bp,ax

pop dx

pop ax

sub al,'0'

sub ah,ah

add bp,ax

mov [di],bp

cmp bp,1000

jl im\_2

jmp im\_4

im\_5:

cmp bp,10

jl im\_3

im\_4:

add di,2

loop im\_1

pop di

pop cx

add di,6\*2

dec bx

jnz im\_0

ret

input\_matrix endp

Для вывода матрицы на экран используются два блока инструкций: output\_matrix и output\_number. Сначала выводится матрица введенная пользователем, потом сортируется и выводится отсортированная матрица.

lea di,matrix1

call output\_matrix ;полный код в приложении 1

Передаем данные в регистр общего назначения и вызываем «функцию» сортировки матрицы.

push bx

push cx

lea di,matrix1

call sort\_matrix

pop cx

pop bx

di – данные

bx – количество строк

sort\_matrix proc

dec bx

mov cx,bx

cmp cx, 0

je sm\_ex

sm\_0:

push cx

push di

mov bx, 1 ; был ли обмен?

sm\_1:

mov ax,[di + 10]

cmp [di + 10 + 12],ax

jae sm\_c

mov si,di

add si,6\*2

call xchange\_lines ; меняем местами

sub bx,bx

sm\_c:

add di,6\*2

loop sm\_1

pop di

pop cx

cmp bx,0

je sm\_0

sm\_ex:

ret

sort\_matrix endp

После сортировки на экран выводится надпись «Sorted matrix:» и отсортированная матрица.

mov ah,09h

lea dx,msg4

int 21h

Вывод сортированной матрицы осуществляется тем же блоком инструкций, что и изначальная матрица.

lea di,matrix1

call output\_matrix

В конце программа выводит на экран сообщение «Done.» и завершает работу.

mov ah,09h

lea dx,msg6 ; Done.

int 21h

mov ax,4c00h ;выход из программы

int 21h **2.** **Результаты тестирования реализованного приложения**

Для тестирования были использованы следующие матрицы:

При тестировании все 3 матрицы были правильно отсортированы. На рисунках 1 и 2 демонстрируется работа программы.

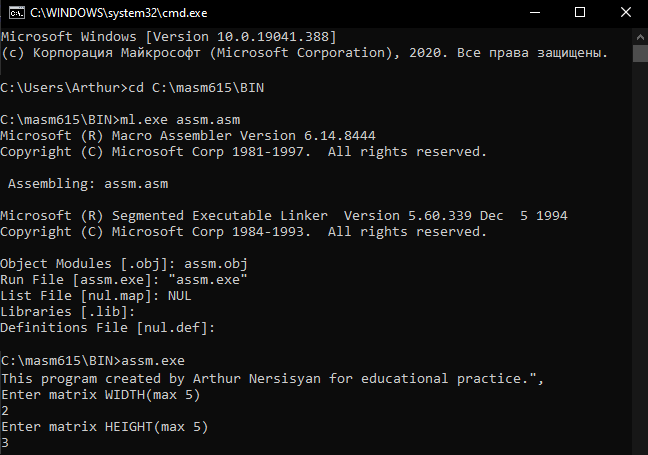


Рисунок 1– Работа программы

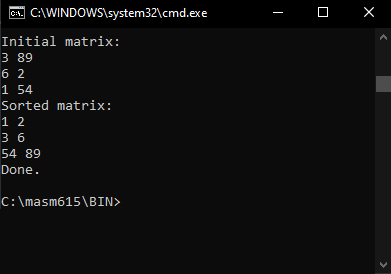


Рисунок 2– Работа программы

**заключение**

В данном проекте была реализована программа, которая сортирует двумерный массив на языке ассемблера (MASM). Программа считывает данные из консоли, анализирует, сортирует и выводит результат на экран. Приложение имеет консольный интерфейс.

**список использованных источников**

1. Калашников О. Ассемблер — это просто. Учимся программировать. — 2-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2011. 336 с.

2. Microsoft Macro Assembler reference // Microsoft Developer Network. URL: https://docs.microsoft.com/en-us/cpp/assembler/masm/microsoft-macro-assembler-reference?view=vs-2019 (дата обращения: 15.07.2020).

**приложение 1**

**Исходный код реализованного приложения на ассемблерЕ**

.386

.model small

.stack 100h

.data

msg1 db "This program created by Arthur Nersisyan for educational practice.",

10, 13, "Enter matrix WIDTH(max 5)", 10, 13, '$'

msg2 db 10, 13, "Enter matrix HEIGHT(max 5)", 10, 13, '$'

msg3 db 10, 13, 10, 13, "Initial matrix: ", 10, 13, '$'

msg4 db 10, 13, "Sorted matrix: ", 10, 13, '$'

msg5 db 10, 13, "Enter number N"

msg5\_w db 0, "x"

msg5\_h db 0, 10, 13, '$'

msg6 db 10, 13, "Done.", 10, 13, '$'

matrix1 dw 30 dup (?)

str\_tmp db 5 dup (' '), '$' ; dlya vyvoda chysel

sum\_sym db "+ ", '$'

equal\_sym db "= ", '$'

next\_line db 10, 13, '$'

.code

start PROC

mov ax,@data

mov ds,ax

call input\_width

call input\_height

; vvod matricy

push bx

push cx

lea di,matrix1

call input\_matrix

pop cx

pop bx

; calculatiya summy

push bx

push cx

lea di,matrix1

call calc\_sum

pop cx

pop bx

mov ah,09h

lea dx,msg3

int 21h

; vyvod nachalnoy matricy

push bx

push cx

lea di,matrix1

call output\_matrix

pop cx

pop bx

; sortirovka

push bx

push cx

lea di,matrix1

call sort\_matrix

pop cx

pop bx

mov ah,09h

lea dx,msg4

int 21h

; vyvod izmenennoy matricy

push bx

push cx

lea di,matrix1

call output\_matrix

pop cx

pop bx

mov ah,09h

lea dx,msg6

int 21h

; vyhod iz programmy

mov ax,4c00h

int 21h

input\_height proc

; bx -> height

mov ah,09h

lea dx,msg2

int 21h

ih\_1:

mov ah,1

int 21h

cmp al,'1'

jl ih\_1

cmp al,'5'

jg ih\_1

sub al,'0'

sub ah,ah

mov bx,ax

ret

input\_height endp

input\_width proc

; cx -> width

mov ah,09h

lea dx,msg1

int 21h

iw\_1:

mov ah,1

int 21h

cmp al,'1'

jl iw\_1

cmp al,'5'

jg iw\_1

sub al,'0'

sub ah,ah

mov cx,ax

ret

input\_width endp

sort\_matrix proc

; di - dannye

; bx - height

dec bx

mov cx,bx

cmp cx, 0

je sm\_ex

sm\_0:

push cx

push di

mov bx, 1 ; byl li obmen?

sm\_1:

mov ax,[di + 10]

cmp [di + 10 + 12],ax

jae sm\_c

; menyaem mestami

mov si,di

add si,6\*2

call xchange\_lines

sub bx,bx

sm\_c:

add di,6\*2

loop sm\_1

pop di

pop cx

cmp bx,0

je sm\_0

sm\_ex:

ret

sort\_matrix endp

xchange\_lines proc

; di - from

; si - to

push cx

push di

push si

mov cx, 6

xl\_1:

mov ax,[di]

mov dx,[si]

mov [si],ax

mov [di],dx

add si,2

add di,2

loop xl\_1

pop si

pop di

pop cx

ret

xchange\_lines endp

calc\_sum proc

; di - dannye

; cx - width

; bx - height

cs\_0:

push cx

push di

sub ax,ax

cs\_1:

add ax,[di]

add di,2

loop cs\_1

pop di

mov [di + 10],ax

pop cx

add di,6\*2

dec bx

jnz cs\_0

ret

calc\_sum endp

output\_number proc

;ax - number

push di

push cx

push si

push dx

lea di,str\_tmp

mov cx,5

on\_1:

;mov [di],' '

inc di

loop on\_1

dec di

dec di

mov si,10

on\_2:

sub dx,dx

div si

add dl,'0'

mov [di],dl

dec di

sub dx,dx

cmp ax,si

jge on\_2

jz on\_3

cmp al,0

jz on\_3

add al,'0'

mov [di],al

on\_3:

mov ah,09h

mov dx,di

int 21h

pop dx

pop si

pop cx

pop di

ret

output\_number endp

output\_matrix proc

; di - dannye

; cx - width

; bx - height

om\_0:

push cx

push di

om\_1:

mov ax,[di]

call output\_number

cmp cx,1

je om\_2

mov ah,09h

lea dx,sum\_sym

int 21h

; 09h вывод строки на экран.На входе - ds:dx = адрес строки с символом $ на конце

; 21h

om\_2:

add di,2

loop om\_1

pop di

mov ah,09h

lea dx,equal\_sym

int 21h

mov ax,[di + 10]

call output\_number

mov ah,09h

lea dx,next\_line

int 21h

pop cx

add di,6\*2

dec bx

jnz om\_0

ret

output\_matrix endp

input\_matrix proc

; di - dannye

; cx - width

; bx - height

mov si,10

sub ax,ax

mov dl,cl

mov dh,bl

im\_0:

push cx

push di

im\_1:

push dx

push di

lea di,msg5\_w

mov al,dl

sub al,cl

add al,'1'

mov [di],al

lea di,msg5\_h

mov al,dh

sub al,bl

add al,'1'

mov [di],al

pop di

mov ah,09h

lea dx,msg5

int 21h

pop dx

sub bp,bp

im\_2:

im\_3: mov ah,1

int 21h

cmp al,0Dh

je im\_5

cmp al,'0'

jl im\_3

cmp al,'9'

jg im\_3

push ax

push dx

mov ax,bp

mul si

mov bp,ax

pop dx

pop ax

sub al,'0'

sub ah,ah

add bp,ax

mov [di],bp

cmp bp,1000

jl im\_2

jmp im\_4

im\_5:

cmp bp,10

jl im\_3

im\_4:

add di,2

loop im\_1

pop di

pop cx

add di,6\*2

dec bx

jnz im\_0

ret

input\_matrix endp

start ENDP

END start