

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 5x5. 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 5x5.

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. 5x5), 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 соответственно.

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 нас 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 выводиться 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

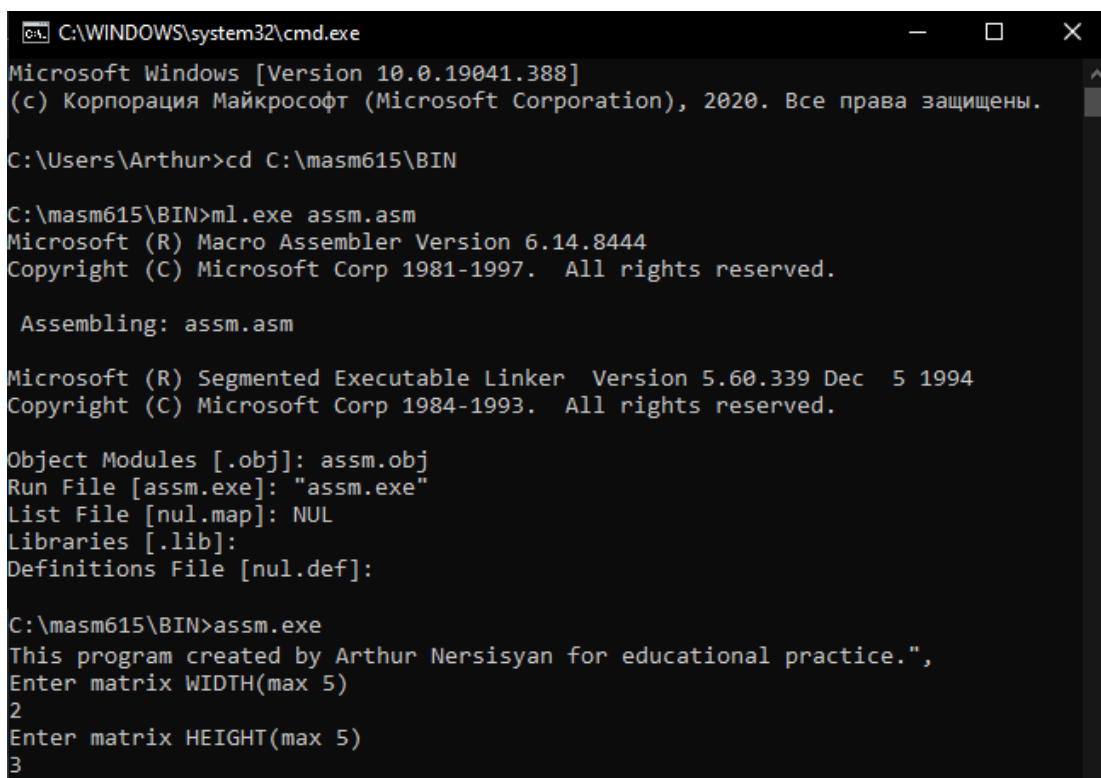
```

2. РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ РЕАЛИЗОВАННОГО ПРИЛОЖЕНИЯ

Для тестирования были использованы следующие матрицы:

$$\begin{pmatrix} 5 & 6 & 4 \\ 48 & 11 & 58 \\ 26 & 81 & 73 \end{pmatrix} \quad \begin{pmatrix} 3 & 89 \\ 6 & 2 \\ 1 & 54 \end{pmatrix} \quad \begin{pmatrix} 62 & 43 & 96 & 74 \\ 35 & 21 & 1 & 63 \\ 85 & 14 & 28 & 65 \\ 27 & 44 & 54 & 78 \end{pmatrix}$$

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



```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 10.0.19041.388]
(c) Корпорация Майкрософт (Microsoft Corporation), 2020. Все права защищены.

C:\Users\Arthur>cd C:\masm615\BIN

C:\masm615\BIN>ml.exe asm.asm
Microsoft (R) Macro Assembler Version 6.14.8444
Copyright (C) Microsoft Corp 1981-1997. All rights reserved.

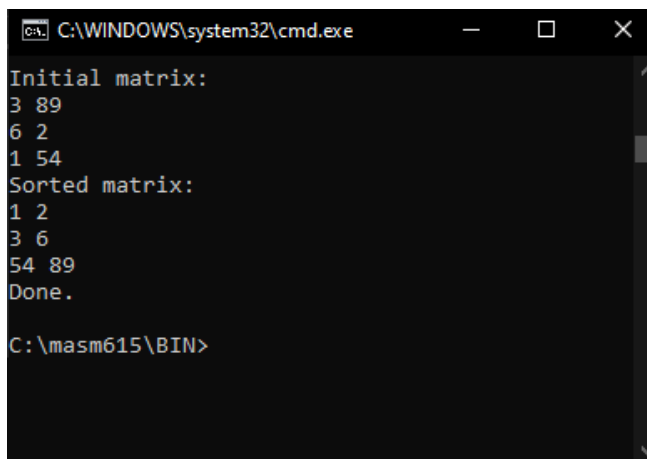
Assembling: asm.asm

Microsoft (R) Segmented Executable Linker Version 5.60.339 Dec 5 1994
Copyright (C) Microsoft Corp 1984-1993. All rights reserved.

Object Modules [.obj]: asm.obj
Run File [asm.exe]: "asm.exe"
List File [nul.map]: NUL
Libraries [.lib]:
Definitions File [nul.def]:

C:\masm615\BIN>asm.exe
This program created by Arthur Nersisyan for educational practice.",
Enter matrix WIDTH(max 5)
2
Enter matrix HEIGHT(max 5)
3
```

Рисунок 1– Работа программы



```
C:\WINDOWS\system32\cmd.exe

Initial matrix:
3 89
6 2
1 54
Sorted matrix:
1 2
3 6
54 89
Done.

C:\masm615\BIN>
```

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/asm/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 - danye
    ; 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 - даные
        ; 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
```