**МИНИСТЕРСТВО ТРАНСПОРТА РОССИЙСКОЙ ФЕДЕРАЦИИ**

**федеральное государственное автономное образовательное**

**учреждение высшего образования**

**«РОССИЙСКИЙ УНИВЕРСИТЕТ ТРАНСПОРТА»**

**(РУТ(МИИТ)**

Институт управления и цифровых технологий

Кафедра «Цифровые технологии управления транспортными процессами»

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №5**

По дисциплине: Ассемблер

На тему: Средства разработки приложений на языке ассемблер

Выполнил студент группы: УИС-312 Рыжов В.Р.

Принял преподаватель кафедры ЦТУТП: Победоносцев И.Н.

**Москва 2022 г.**

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc118635765)

[Поставленные цели 3](#_Toc118635766)

[Поставленные задачи 3](#_Toc118635767)

[1 БЛОК-СХЕМА 4](#_Toc118635768)

[2 ТЕКСТ ПРОГРАММЫ С КОМЕНТАРИЯМИ 4](#_Toc118635769)

[3 РЕЗУЛЬТАТЫ РАБОТЫ ПРОГРАММЫ 6](#_Toc118635770)

[4 СОДЕРЖИМОЕ СТЕКА 7](#_Toc118635771)

[5 ВОПРОСЫ 9](#_Toc118635772)

[ЗАКЛЮЧЕНИЕ 10](#_Toc118635773)

# ВВЕДЕНИЕ

Процедура представляет собой группу команд для решения конкретной подзадачи. Процедура - именованная, правильным образом оформленная группа команд, которая, будучи однократно описана, при необходимости может быть вызвана по имени любое количество раз из различных мест программы. Описание процедуры. Для описания последовательности команд в виде процедуры в языке ассемблера используются две директивы: PROC и ENDP.

## Поставленные цели

В цели работы входит изучение:

* структуры стека и приемов его использования;
* команд вызова и возврата из процедур;
* способа передачи параметров при вызове процедур с использованием стека;
* написание и отладка программ, использующих процедуры.

## Поставленные задачи

1. Добавить процедуру в программу, написанную в четвертой лабораторной работе.
2. Проверить работоспособность программы и полученные на тестовом примере результаты. Подготовить для отчета исходную матрицу А и результирующий одномерный массив B в десятичном формате, а также в виде скриншота дампа сегмента данных.
3. C помощью отладчика проследить за изменением состояния стека в ходе выполнения программы

# 1 БЛОК-СХЕМА

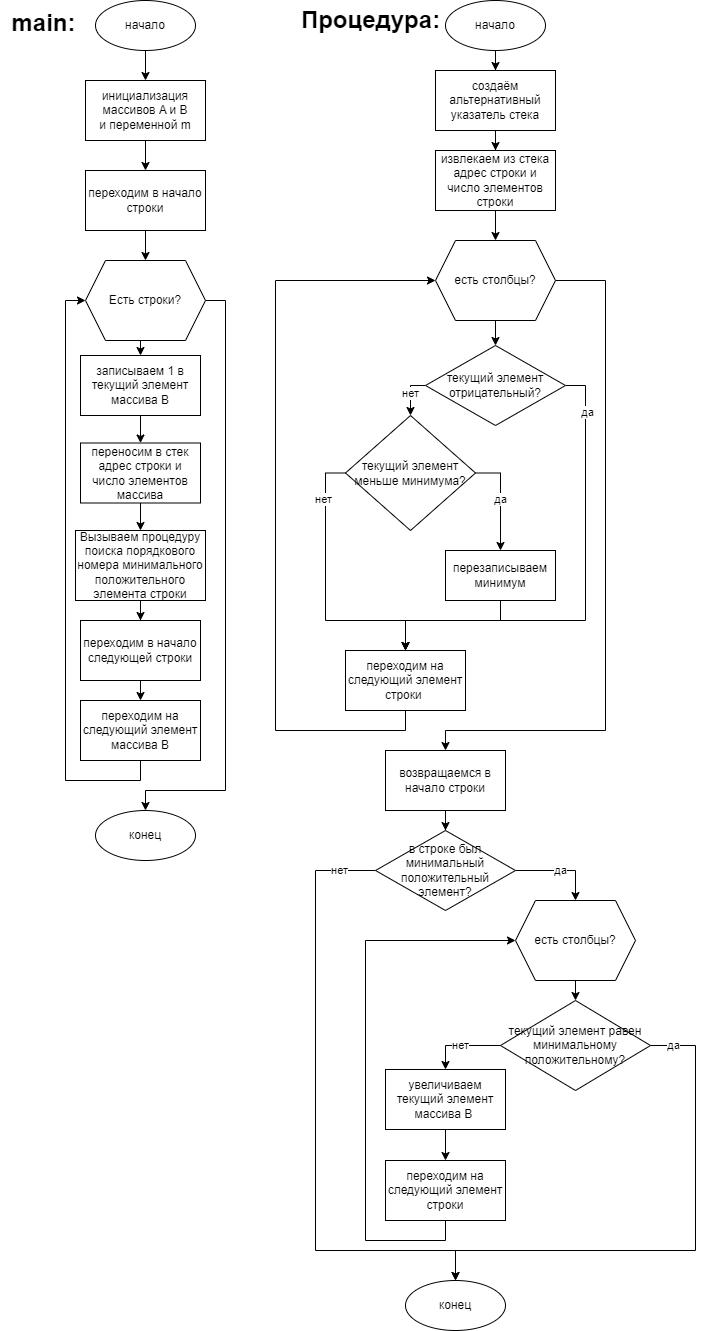


Рисунок 1.1 – блок-схема алгоритма

# 2 ТЕКСТ ПРОГРАММЫ С КОМЕНТАРИЯМИ

;Лабораторная работа №5

;Выполнил ст. гр. УИС-312.

;Вариант 20

;Дан двумерный массив А(4х5).

;Сформировать одномерный массив B(4),каждый элемент которого

;Порядковый номер минимального положительного элемента (или – 1)

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

.486

.model flat, stdcall

option casemap: none

.stack 100h

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

include <\masm32\include\kernel32.inc>

includelib <\masm32\lib\kernel32.lib>

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

.data

N equ 4 ;количество строк

M equ 5 ;количество столбцов

A db 1,2,-3,-4,5

db 6,7,-8,9,0

db -1,-3,-2,-7,-4

db 6,-8,2,0,1

B db 4 dup (?)

m db 255

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

.code

;=====Процедура================================================

;Процедура поиска порядкового номера минимального положительного элемента

;Входные параметры (передаются через стек):

; - адрес начала массива [ebp+8]

; - количество элементов массива [ebp+12]

;Выходной параметр (передается через регистр al - сумма элементов строки)

checkstr proc

push ebp ;спасаем ebp

mov ebp,esp ;формирование альтернативного указателя стека ebp=esp

push ecx ;спасаем используемые в процедуре регистры

push ebx

mov ecx,[ebp+8] ;извлечение из стека параметра 2 (число элементов массива)

mov ebx,[ebp+12];извлечение из стека параметра 1 (адреса массива)

;-----------------

next: ;начало цикла по столбцам (внутренний цикл)

mov al, [ebx]; берём первый элемент, как минимальный

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

cmp al, 0 ; сравниваем m с нулем

jl skip; если число отрицательное, то пропускаем его

cmp al, m

JAE skip; если число больше или равно, то пропускаем его

mov m,al

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

skip:

inc ebx

loop next ;продолжить, если выбраны не все эл-ты строки

;строки массива A

;-----------------

mov ecx,M

sub ebx,5

cmp m,255

Je exit

search: ; цикл поиска номера минимального положительного элемента

mov al,[ebx]

cmp al,m

je exit

add B[edi],1

inc ebx

loop search

exit:

mov m, 255

;-----------------

pop ebx ;восстановить спасаемые регистры

pop ecx

pop ebp

ret 8 ;зачистить стек от параметров и возврат из процедуры

checkstr endp

;==== Основная программа=======================================

main:

;для адресации строк массива A используем регистр

; ebx - адрес (смещение) текущей строки

;для адресации элементов массива B используем регистр

; edi - смещение элемента

mov ecx,N ;счетчик строк массива A

mov edx,M ;количество элементов в строке

mov edi,0 ;смещение эл-та массива B

lea ebx,A ;адрес первой строки массива A

;-----------------

stroka: ;начало цикла по строкам А

;записываем в стек входные параметры для процедуры:

push ebx ;параметр 1: адрес выбранной строки массива A

push edx ;параметр 2: число элементов в строке массива A

mov B[edi],1 ;обнуляем текущий элемент массива B

call checkstr ;вызов процедуры суммирования

;-----------------

add ebx,M\*type A ;установить ebx на начало следующей строки A (+5)

add edi,type B ;установить edi на следующий элемент массива B(+1)

loop stroka ;продолжить, если обработаны не все строки

;-----------------

invoke ExitProcess, 0 ;завершение программы c кодом 0

end main

# 3 РЕЗУЛЬТАТЫ РАБОТЫ ПРОГРАММЫ

На приведенном ниже рисунке 3.1 виден результат работы программы.

* Красный прямоугольник выделяет полученный в ходе работы массив

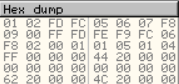


рисунок 3.1 – результат работы программы

На таблице 3.2-3.3 виден исходный массив А и полученный массив B.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | -3 | -4 | 5 |
| 6 | 7 | -8 | 9 | 0 |
| -1 | -3 | -2 | -7 | -4 |
| 6 | -8 | 2 | 0 | -1 |

рисунок 3.2 – массив А

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 5 | 1 | 4 |

рисунок 3.3 массив B

# 4 СОДЕРЖИМОЕ СТЕКА

В рисунке 4.1-4.3 показаны результаты заполнения и очистки стека для первого цикла. На таблице 4.4 виден процесс заполнения и очистки стека во время работы программы.



рисунок 4.1 – обращение к стеку



рисунок 4.2 – в стек внесены данные



рисунок 4.3 – стек очищен

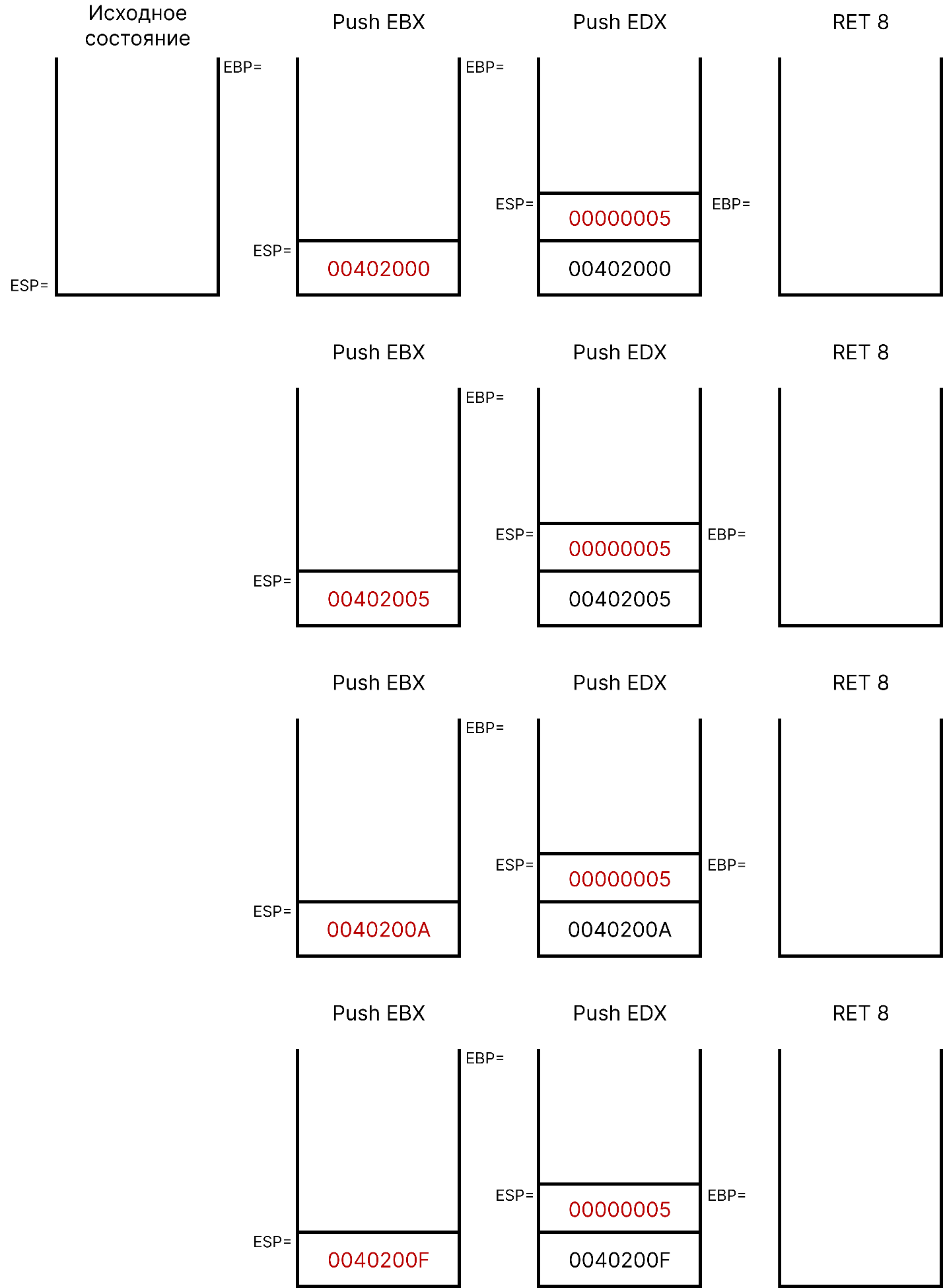


таблица 4.4 – процесс работы стека

# 5 ВОПРОСЫ

1. С какой целью используют процедуры?

* Декомпозиция

1. Что такое стек? Поясните работу команд PUSH и POP.

* Для хранения адреса возврата, параметров и локальных переменных процедуры используется специальная область памяти программы - стек, определяемая в виде отдельного сегмента с помощью директивы: .STACK операнд;
* Программа может поместить информацию в стек с помощью команды PUSH или извлечь ее из стека с помощью команда POP.
* POP операнд. Команда POP извлекает из стека четырех- или двухбайтовое содержимое по адресу ESP , копирует извлеченное значение в указанный операнд и формирует новое значение указателя стека.
* PUSH операнд. Команда PUSH заносит в стек четырех- или двухбайтовый операнд, изменяя при этом текущее значение указателя стека ESP.

1. Почему команду CALL называют "переход с возвратом"? Как обеспечивается возврат в точку вызова?

* CALL имя\_процедуры Команда CALL производит безусловный переход на метку имя\_процедуры, предварительно сохраняя в стеке текущее значение указателя команд EIP. Вспомним, что EIP всегда содержит адрес следующей команды. В данном случае это будет адрес команды, следующей за командой CALL. Его называют адресом возврата из процедуры, поскольку после завершения процедуры должен быть обеспечен автоматический переход по этому адресу.

1. Каков алгоритм выполнения команды RET.

* RET [число] Команда RET извлекает из стека адрес возврата, ранее записанный туда командой CALL, и обеспечивает автоматическое продолжение выполнения программы, начиная с команды, следующей за командой CALL. Необязательный параметр число позволяет дополнительно удалить из стека заданное количество байт (обычно используется для очистки стека от размещенных в нем параметров процедуры).

1. Что произойдет при выполнении программы, приведенной в п. 6.8, если в процедуре SUMSTR будет отсутствовать команда RET?

* Не будет очищен стек.

1. Можно ли передавать управление в процедуру с помощью команды JMP? Что произойдет?

* JMP передает управление в другую точку программы, не сохраняя какой-либо информации для возврата. Операндом может быть непосредственный адрес для перехода (в программах используют имя метки, установленной перед командой, на которую выполняется переход), а также регистр или переменная, содержащая адрес.

1. В стек последовательно были записаны двухбайтовые числа 1, 2, 3, 4, 5. Какое значение примет регистр EAX после выполнения следующих команд в указанной последовательности: PUSH EAX POP EAX POP EBX POP ECX POP EAX

- Последовательность PUSH EAX POP EAX ничего не делает со стеком,

затем в EBX записывается 5 и указатель стека смещается на 4 бита,

затем в ЕСХ записывается 3 и указатель стека смещается на 4 бита,

и в ЕАХ окажется 1.

# ЗАКЛЮЧЕНИЕ

В ходе данной работы я научился работать с процедурами в коде для ассемблера, которые позволяют избежать дублирования кода и значительно уменьшают его общую длину. А также лучше понял алгоритмы работы программ, написанных для ассемблера.