*Федеральное государственное бюджетное образовательное учреждение* *высшего образования*

|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | ***«Московский государственный технический университет  имени Н.Э. Баумана***  ***(национальный исследовательский университет)»***  ***(МГТУ им. Н.Э. Баумана)*** |

ФАКУЛЬТЕТ \_ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ\_\_\_\_\_\_\_\_\_\_\_\_\_

КАФЕДРА \_\_\_\_КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Отчет**

**по лабораторной работе № \_**4**\_\_\_**

**Дисциплина: \_**Машинно-зависимые языки и основы компиляции**\_\_\_\_**

**Название лабораторной работы: \_**Обработка массивов и матриц**\_\_\_\_\_**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Студент гр. **\_**ИУ6-42б**\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_**И.С. Марчук**\_\_\_**

(Подпись, дата) (И.О. Фамилия)

Преподаватель  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

(Подпись, дата) (И.О. Фамилия)

Москва, 2021

**Задание:**

Разработать программу на языке ассемблера, которая решает поставленную задачу.

Вариант 16:Дана матрица 5х5. Определить на каждой строке произведение элемент, кратных 4, и поместить его на место элементов побочной диагонали. Организовать ввод матрицы и вывод результатов.

**Цель работы:**

Изучение приемов моделирования обработки массивов и матриц в языке ассемблера.

**Схема алгоритма для решения поставленной задачи представлена на рисунках 1-3:**

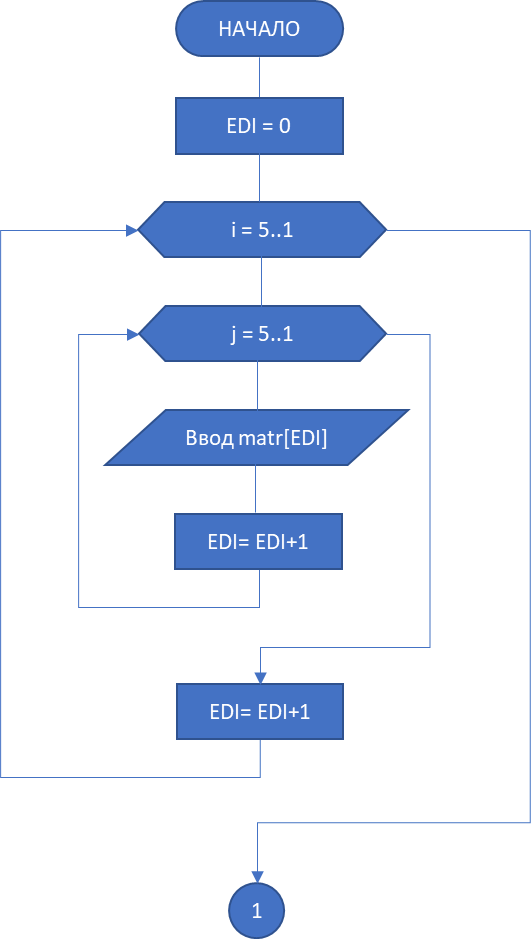


Рисунок 1 – Первая часть схемы алгоритма (Ввод матрицы)

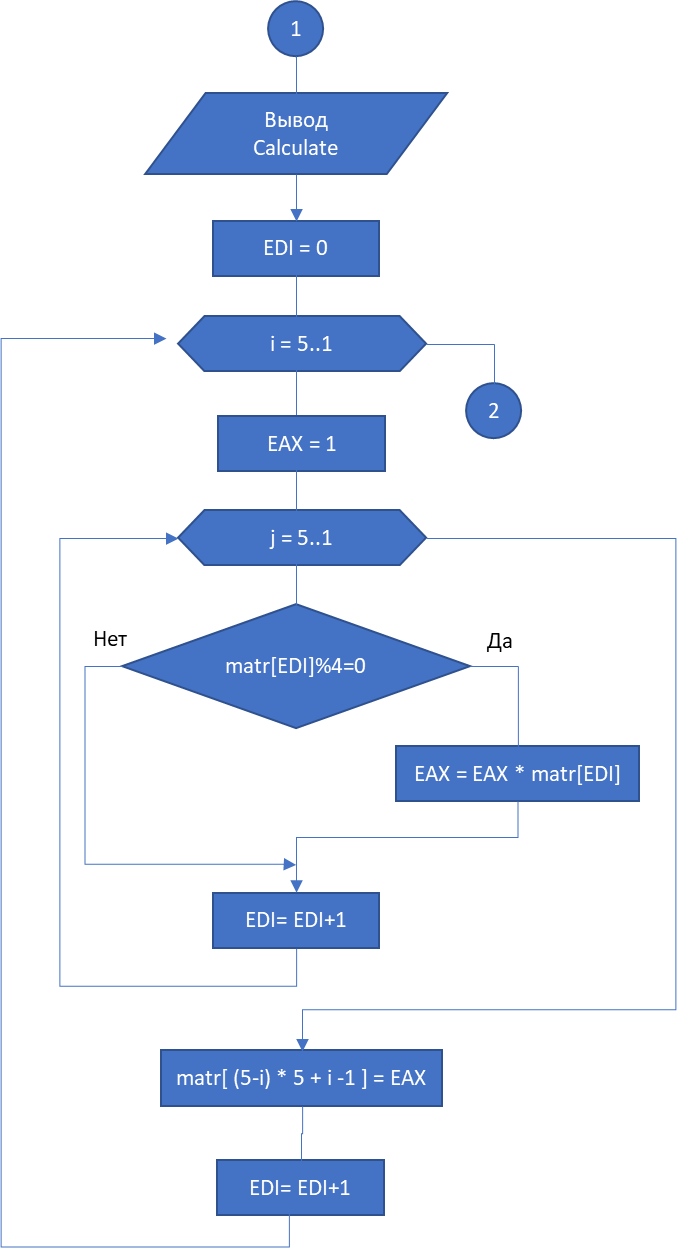


Рисунок 2 – Вторая часть схемы алгоритма (Расчеты)

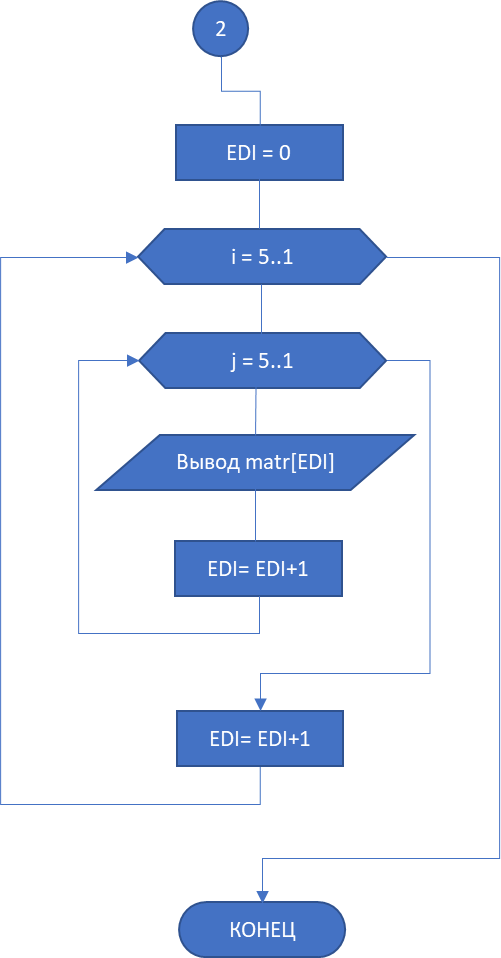


Рисунок 3 – Заключительная часть схемы алгоритма (Вывод матрицы)

**Текст программы с комментариями:**

**.586** ; подключение набора команд Реntium

**.MODEL flat, stdcall** ; модель памяти и

; конвенция о передаче параметров

**OPTION CASEMAP:NONE** ; опция различия строчных

; и прописных букв

**Include kernel32.inc** ; подключение описаний процедур и

**Include masm32.inc** ; констант

**IncludeLib kernel32.lib** ; подключение библиотек

**IncludeLib masm32.lib**

**.CONST**

**msgEnterLine DB "Enter 5 values - line ",0**

**msgCalculate DB "Calculate",0AH,0DH,0**

**msgExit DB "Press Enter to Exit",0AH,0DH,0**

**nextLine DB 0AH,0DH,0**

**space DB " ",0**

**.DATA**

;matr DWORD 1,2,3,4,5,6,7,8,9,10,1,2,3,4,5,6,7,8,9,20,1,2,3,4,5

**.DATA?**

**matr DWORD 25 DUP (?)**

**myInOutBuffer DB 10 DUP (?)**

**regOutBuf DWORD ?**

**.CODE**

**Start:**

; ввод матрицы matr 5х5

**mov ECX, 5** ; Количество повторений цикла

**mov EDI, 0** ; Индекс элемента

**cycle\_str\_in:** ; Цикл по строкам

**push ECX**

**MOV regOutBuf, ECX**

**Invoke StdOut, ADDR msgEnterLine** ; Выводим приглашение новой строки

**Invoke dwtoa, regOutBuf, ADDR myInOutBuffer** ; Переводим элемент в строку

**Invoke StdOut, ADDR myInOutBuffer** ; Выводим элемент

**Invoke StdOut, ADDR nextLine** ; вывод переноса

**mov ECX, 5** ; Количество повторений цикла

**cycle\_col\_in:** ; Цикл по столбцам

**push ECX** ; Помещаем значение ECX в стэк

**Invoke StdIn, ADDR myInOutBuffer, LengthOf myInOutBuffer** ; вводим элемент

**Invoke StripLF, ADDR myInOutBuffer** ; ноль в конец

**Invoke atol, ADDR myInOutBuffer** ; переводим строку с нулем в число

**MOV matr[EDI\*4], EAX**

**inc EDI** ; Увеличиваем индекс элемента на 1

**pop ECX** ; Берем значение ECX из стэка

**loop cycle\_col\_in**

**pop ECX**

**loop cycle\_str\_in**

; расчеты

**Invoke StdOut, ADDR msgCalculate**

**mov ECX, 5** ; Количество повторений цикла

**mov EDI, 0 ;** Индекс элемента

**cycle\_str\_calc:** ; Цикл по строкам

**push ECX**

**mov EAX, 1** ; изначальное произведение всех элементов в строке

**mov ECX, 5** ; Количество повторений цикла

**cycle\_col\_calc:** ; Цикл по столбцам

**push ECX** ; Помещаем значение ECX в стэк

**push EAX**

**mov EAX, matr[EDI\*4];** проверка кратности 4

**mov EDX, 0**

**mov EBX, 4**

**div BX**

**pop EAX**

**cmp DX, 0**

**JNE calc\_else** ; если не кратно, пропускаем следующий шаг

**mul matr[EDI\*4]** ; если кратно, умножаем общее произведение

**calc\_else:**

**inc EDI** ; Увеличиваем индекс элемента на 1

**pop ECX** ; Берем значение ECX из стэка

**loop cycle\_col\_calc**

**pop ECX**

**PUSH EAX**

**MOV EAX, 5** ;расчитываем позицию побочной диагонали

**SUB EAX, ECX**

**MOV EBX, 5**

**MUL EBX**

**ADD EAX, ECX**

**SUB EAX, 1**

**MOV EBX, EAX**

**POP EAX**

**MOV matr[EBX\*4], EAX** ; записываем получившееся значение на побочной диагонали массива

**LOOP cycle\_str\_calc**

; вывод

**mov ECX, 5** ; Количество повторений цикла

**mov EDI, 0** ; Индекс элемента

**cycle\_str\_output:** ; Цикл по строкам

**push ECX**

**mov ECX, 5** ; Количество повторений цикла

**cycle\_col\_output:** ; Цикл по столбцам

**push ECX** ; Помещаем значение ECX в стэк

**Invoke dwtoa, matr[EDI\*4], ADDR myInOutBuffer** ; Переводим элемент в строку

**Invoke StdOut, ADDR myInOutBuffer** ; Выводим элемент

**Invoke StdOut, ADDR space** ; Вывдом символ разделения

**inc EDI** ; Увеличиваем индекс элемента на 1

**pop ECX** ; Берем значение ECX из стэка

**loop cycle\_col\_output**

**Invoke StdOut, ADDR nextLine** ; Вывдом символ новой строки

**pop ECX**

**loop cycle\_str\_output**

**XOR EAX,EAX**

**Invoke StdOut,ADDR msgExit**

**Invoke StdIn,ADDR myInOutBuffer,LengthOf myInOutBuffer**

**Invoke ExitProcess,0**

**End Start**

**Работа программы с тестовыми данными (таблица 1 и рисунок 4):**

|  |  |  |
| --- | --- | --- |
| Исходные данные | Ожидаемый результат | Полученный результат |
| 1 2 3 4 5  6 7 8 9 10  11 12 13 14 15  16 17 18 19 20  21 22 23 24 25 | 1 2 3 4 **4**  6 7 8  **8** 10  11 12 **12** 14 15  16 **320** 18 19 20  **24** 22 23 24 25 | 1 2 3 4 4  6 7 8 8 10  11 12 12 14 15  16 320 18 19 20  24 22 23 24 25 |
| 1 2 3 4 5  6 7 8 9 10  1 2 3 4 5  6 7 8 9 20  1 2 3 4 5 | 1 2 3 4  **4**  6 7 8  **8** 10  1 2 **4** 4 5  6  **160** 8 9 20  **4**  2 3 4 5 | 1 2 3 4 4  6 7 8 8 10  1 2 4 4 5  6 160 8 9 20  4 2 3 4 5 |
| 1 5 3 7 4  7 5 6 5 5  3 3 3 3 3  4 4 4 4 4  1 1 1 1 1 | 1 5 3 7 **4**  7 5 6 **1** 5  3 3 **1** 3 3  4 **1024** 4 4 4  **1** 1 1 1 1 | 1 5 3 7 4  7 5 6 1 5  3 3 1 3 3  4 1024 4 4 4  1 1 1 1 1 |
| 0 2 5 4 2  0 1 2 3 3  4 5 2 0 9  4 1 2 3 6  3 6 9 2 6 | 0 2 5 4 **0**  0 1 2 **0** 3  4 5 0 0 9  4 **4** 2 3 6  **1** 6 9 2 6 | 0 2 5 4 0  0 1 2 0 3  4 5 0 0 9  4 4 2 3 6  1 6 9 2 6 |
| 33 -36 -48 35 -26  78 -13 51 60 -1  41 26 79 0 -79  -74 -2 -50 -46 -14  90 -69 45 32 88 | 33 -36 -48 35 **1728**  78 -13 51 **60** -1  41 26 **0** 0 -79  -74 **1** -50 -46 -14  **2816** -69 45 32 88 | 33 -36 -48 35 1728  78 -13 51 60 -1  41 26 0 0 -79  -74 1 -50 -46 -14  2816 -69 45 32 88 |
| 48 9 -54 90 -16  -74 87 61 -85 -25  -23 -3 15 -18 -4  22 35 -80 -15 -7  -29 -90 72 89 41 | 48 9 -54 90 **-768**  -74 87 61 **1** -25  -23 -3 **-4** -18 -4  22 **-80** -80 -15 -7  **72** -90 72 89 41 | 48 9 -54 90 -768  -74 87 61 1 -25  -23 -3 -4 -18 -4  22 -80 -80 -15 -7  72 -90 72 89 41 |
| -6 0 -8 7 9  -6 -6 -7 1 -7  -8 3 -1 9 8  -8 7 8 8 -8  3 1 -2 -8 -5 | -6 0 -8 7 **0**  -6 -6 -7 **1** -7  -8 3 **-64** 9 8  -8 **4096** 8 8 -8  **-8** 1 -2 -8 -5 | -6 0 -8 7 0  -6 -6 -7 1 -7  -8 3 -64 9 8  -8 4096 8 8 -8  -8 1 -2 -8 -5 |
| -7 0 -4 3 -9  -3 0 -6 6 2  1 1 -1 -1 3  -8 4 0 -3 -2  2 3 -9 3 1 | -7 0 -4 3 **0**  -3 0 -6 **0** 2  1 1 **1** -1 3  -8 **0** 0 -3 -2  **1** 3 -9 3 1 | -7 0 -4 3 0  -3 0 -6 0 2  1 1 1 -1 3  -8 0 0 -3 -2  1 3 -9 3 1 |

Таблица 1 – Отладка программы

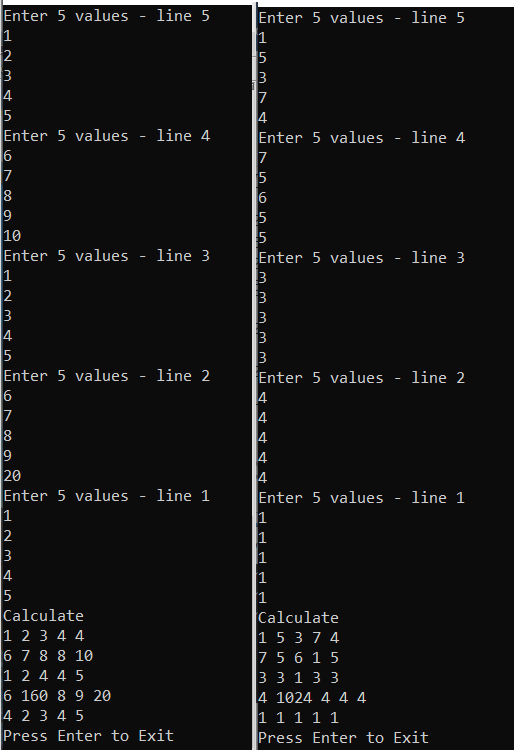


Рисунок 4 – Вывод программы при различных входных данных

**Контрольные вопросы:**

1) Почему в ассемблере не определены понятия «массив», «матрица»?

Т.к. ассемблер это язык низкого уровня, в котором операнды – регистры и память. Ассемблер обращается напрямую к памяти, а в памяти нет структур таких как массивы. В ассемблере массивы можно задать только как концепт, на деле же, это будет упорядоченные чтение и запись напрямую в память.

1. Как в ассемблере моделируются массивы?

В ассемблере массивы можно представить как ряд повторяющихся данных в памяти, это например строки или переменные определенные как повторяющиеся:

**CONST**

**msgEnterLine DB "Enter 5 values - line ",0**

**msgCalculate DB "Calculate",0AH,0DH,0**

**msgExit DB "Press Enter to Exit",0AH,0DH,0**

**nextLine DB 0AH,0DH,0**

**space DB " ",0**

**.DATA**

**matr DWORD 1,2,3,4,5,6,7,8,9,10,1,2,3,4,5,6,7,8,9,20,1,2,3,4,5**

**.DATA?**

**matr DWORD 25 DUP (?)**

**myInOutBuffer DB 10 DUP (?)**

Все представленное выше имеет структуру массивов.

3) Поясните фрагмент последовательной адресации элементов массива? Почему при этом для хранения частей адреса используют регистры?

Нельзя производить операции между памятью и памятью, один из операндов всегда должен быть в регистре поэтому элементы в программе последовательно выгружаются в регистры. Также позицию данных в памяти необходимо вычислять, а потому адрес удобнее всего хранить в регистрах.

4) Как в памяти компьютера размещаются элементы матриц?

Последовательно, строка за строкой.

5) Чем моделирование матриц отличается от моделирования массивов? В каких случаях при выполнении операций для адресации матриц используется один регистр, а в каких – два?

В матрицах следует различать просмотр по строкам, просмотр по столбцам, просмотр по диагоналям и произвольный доступ. Если матрица расположена в памяти по строкам и просмотр выполняется по строкам, то обработка может выполняться так, как в одномерном массиве, без учета перехода от одной строки к другой. Просмотр по строкам при необходимости фиксировать завершение строки и просмотр по столбцам при построчном расположении в памяти выполняются в двойном цикле. При просмотре по диагонали обычно используют один цикл, через переменную которого рассчитываются смещения элементов массива. Однако проще использовать специальный регистр смещения, который должен соответствующим образом переадресовываться.

**Вывод:**

Я изучил приемы моделирования обработки массивов и матриц в языке ассемблера. Определил их способы хранения в памяти, и различия способов работы с ними. И запрограммировал решение поставленной задачи.