Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«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)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.01 Информатика и вычислительная техника**

ОТЧЕТ

По лабораторной работе №4

Название : Программирование обработки массивов и матриц

Дисциплина: Машинно-зависимые языки и основы компиляции

Студент	ИУ-42б	_____	С.В. Астахов
	(Группа)	(Подпись, дата)	(И.О. Фамилия)
Преподаватель		_____	_____
		(Подпись, дата)	(И.О. Фамилия)

1 вариант

Москва, 2021

Задание

Дана матрица 7x3. Обнулить элементы матрицы с четной суммой индексов. Организовать ввод матрицы и вывод результатов.

Текст программы:

```
; Template for console application
         .586
         .MODEL  flat, stdcall
         OPTION CASEMAP:NONE

Include kernel32.inc
Include masm32.inc

IncludeLib kernel32.lib
IncludeLib masm32.lib

         .CONST
MsgExit  DB    "Press Enter to Exit",0AH,0DH,0
reqMatr DB 'Input num: ',13,10,0
reqHello  DB  '===  Input  21  num  for  7x3  matrix  [0..6,  0..2]
===',13,10,0
reqEcho DB '=== Echo of your input [0..6, 0..2] ===',13,10,0
reqResult DB '=== Result [0..6, 0..2] ===',13,10,0
MsgLn DB 0AH,0DH,0
MsgSpace DB "    ",0
RowLg WORD 3
two_sizes WORD 8 ;=2*4bytes
_3 WORD 3
X SDWORD 4

         .DATA
Matr SDWORD 25 DUP (7)
log db 20 dup (?)
         .DATA?
Cnt byte ?
MsgResult DB 10 DUP (?)
inbuf    DB    100 DUP (?)
buffer DB 10 DUP (?)
outstr DB 10 DUP (?)
A SWORD ?
B SWORD ?
i SDWORD ?
j SDWORD ?
res SDWORD ?
```

```
        .CODE
Start:
;
;    Add you statements
;
        XOR    EAX,EAX

        Invoke StdOut,ADDR reqHello

;=================== input cycle ================

         mov i,0
         mov j,0
         alt_out_cycle0:
             mov j,0
             ;Invoke StdOut,ADDR MsgLn
             alt_in_cycle0:

               Invoke StdOut,ADDR reqMatr
               Invoke StdIn,ADDR buffer,LengthOf buffer
               Invoke StripLF,ADDR buffer
               Invoke atol,ADDR buffer ;result in EAX

               mov EBX,i
               mov ECX,j

               mov Matr[EBX+ECX], EAX
               ;Invoke dwtoa,Matr[EBX+ECX],ADDR outstr
               ;Invoke StdOut,ADDR outstr
               ;Invoke StdOut,ADDR MsgSpace
               add j,4
               cmp j,12
             jl alt_in_cycle0
           add i,12
           cmp i,84
           jl alt_out_cycle0

;=========== echo cycle ============================

        Invoke StdOut,ADDR MsgLn
        Invoke StdOut,ADDR reqEcho
        Invoke StdOut,ADDR MsgLn

         mov i,0
         mov j,0
         alt_out_cycle:
             mov j,0
             Invoke StdOut,ADDR MsgLn
             alt_in_cycle:
               mov EBX,i
```

```
                mov ECX,j
                Invoke dwtoa,Matr[EBX+ECX],ADDR outstr
                Invoke StdOut,ADDR outstr
                Invoke StdOut,ADDR MsgSpace
                add j,4
                cmp j,12
              jl alt_in_cycle
            add i,12
            cmp i,84
            jl alt_out_cycle

          Invoke StdOut,ADDR MsgLn
;========== deleting cycle =========

        mov i,0
        mov j,0
          alt_out_cycle1:
              mov j,0
              ;Invoke StdOut,ADDR MsgLn
              alt_in_cycle1:

                mov EBX,i
                mov ECX,j
                ;mov EAX,0
                mov EAX, i


                cwd ;DX:AX = AX
                idiv _3;AX:=(DX:AX):2   DX = remain
                add EAX,j
                cwd ;DX:AX = AX

                ;mov res, EAX

                idiv two_sizes;AX:=(DX:AX):2   DX = remain
                cmp DX,0
                jne odd
                    mov Matr[EBX+ECX], 0
                odd: ; x%2 = 1

                ;Invoke dwtoa,res,ADDR log
                ;Invoke StdOut,ADDR MsgLn
                ;Invoke StdOut,ADDR log

                add j,4
                cmp j,12
              jl alt_in_cycle1
            add i,12
            cmp i,84
            jl alt_out_cycle1


;=========== output result ============
```

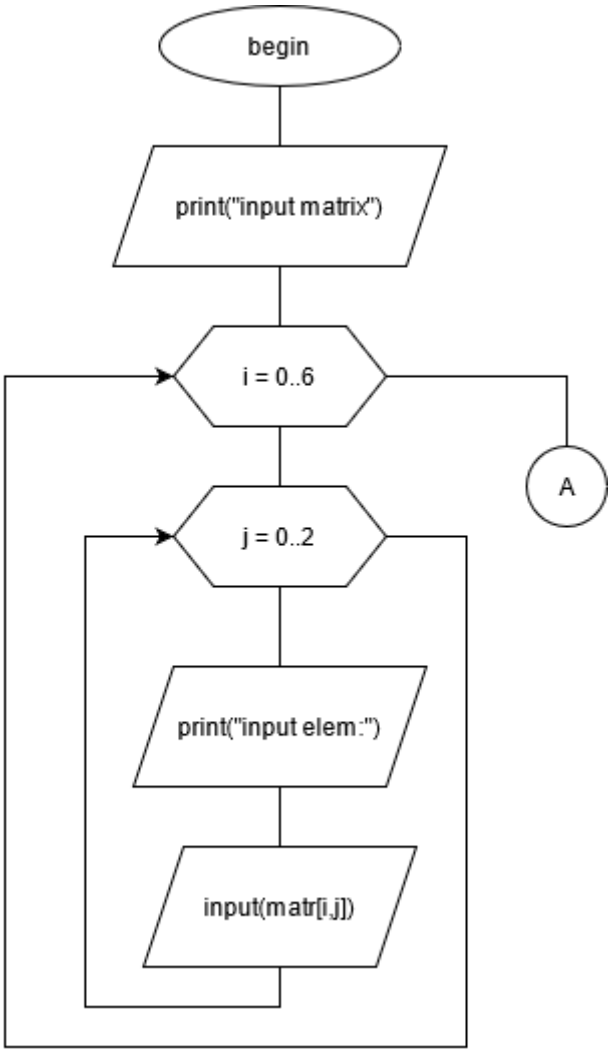
```
Invoke StdOut,ADDR MsgLn
Invoke StdOut,ADDR reqResult
Invoke StdOut,ADDR MsgLn

mov i,0
 mov j,0
 alt_out_cycle2:
     mov j,0
     Invoke StdOut,ADDR MsgLn
     alt_in_cycle2:
       mov EBX,i
       mov ECX,j
       Invoke dwtoa,Matr[EBX+ECX],ADDR outstr
       Invoke StdOut,ADDR outstr
       Invoke StdOut,ADDR MsgSpace
       add j,4
       cmp j,12
     jl alt_in_cycle2
   add i,12
   cmp i,84
   jl alt_out_cycle2

 Invoke StdOut,ADDR MsgLn


Invoke StdOut,ADDR MsgLn
Invoke StdOut,ADDR MsgExit
Invoke StdIn,ADDR inbuf,LengthOf inbuf

Invoke ExitProcess,0
End    Start
```

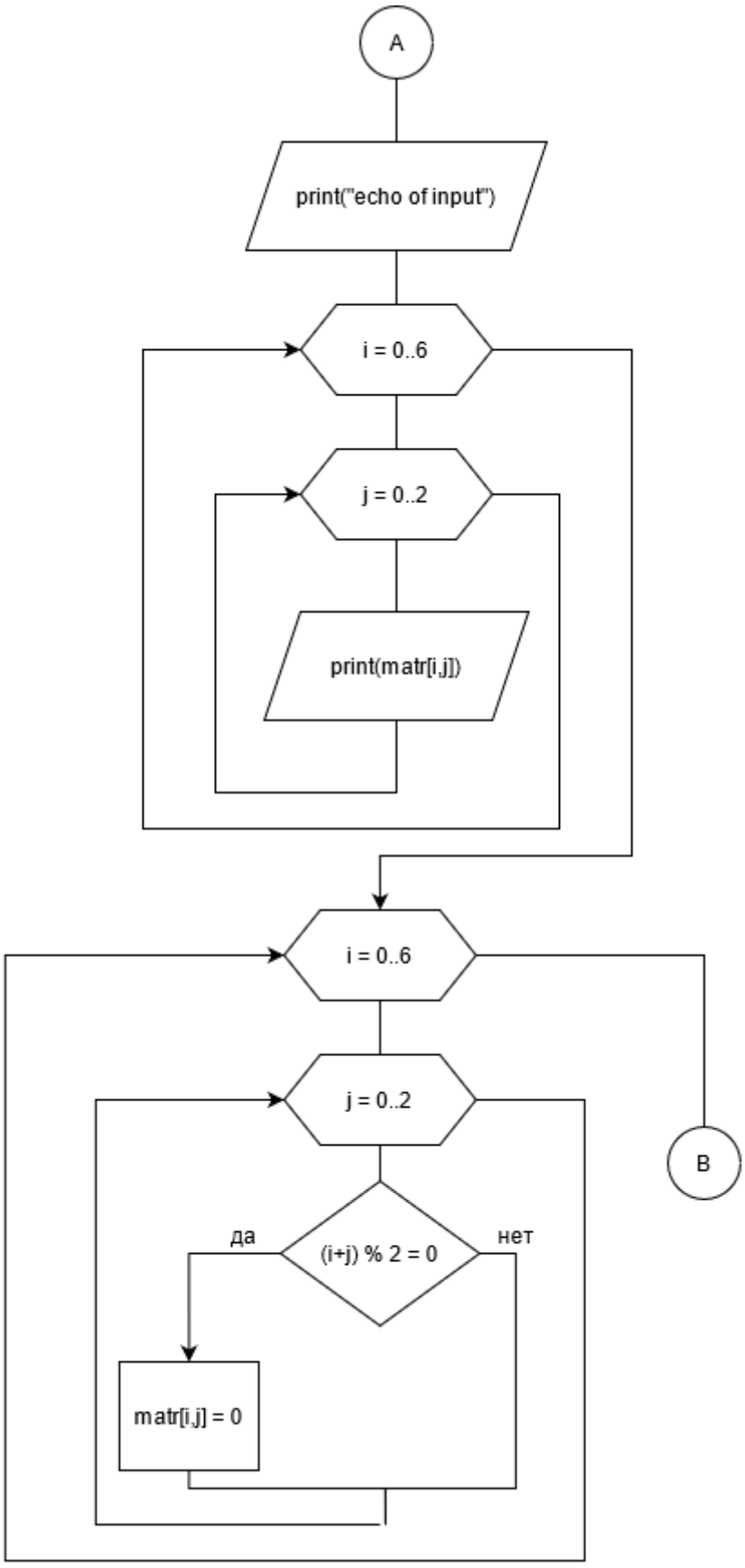
Таблица 1 - Результаты тестирования

Исходные данные	Ожидаемый результат	Полученный результат
7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7	0 7 0 7 0 7 0 7 0 7 0 7 0 7 0 7 0 7 0 7 0	0 7 0 7 0 7 0 7 0 7 0 7 0 7 0 7 0 7 0 7 0
1 -2 -3 4 5 1 -2 -3 4	0 -2 0 4 0 1 0 -3 0	0 -2 0 4 0 1 0 -3 0

5 1 -2 -3 4 5 1 -2 -3 4 5 1	5 0 -2 0 4 0 1 0 -3 0 5 0	5 0 -2 0 4 0 1 0 -3 0 5 0
-21 -21 -21 -21 -21 -21 -21 -21 -21 -21 -21 -21 -21 -21 -21 -21 -21 -21 -21 -21 -21	0 -21 0 -21 0 -21 0 -21 0 -21 0 -21 0 -21 0 -21 0 -21 0 -21 0	0 -21 0 -21 0 -21 0 -21 0 -21 0 -21 0 -21 0 -21 0 -21 0 -21 0
1 -45 -2 3 5 15 48 1 -45 -2 3 5 15 48 1 -45 -2 3 5 15 48	0 -45 0 3 0 15 0 1 0 -2 0 5 0 48 0 -45 0 3 0 15 0	0 -45 0 3 0 15 0 1 0 -2 0 5 0 48 0 -45 0 3 0 15 0

Схема алгоритма:




A
print("echo of input")
i = 0..6
j = 0..2
print(matr[i,j])
i = 0..6
j = 0..2
B
да
(i+j) % 2 = 0
нет
matr[i,j] = 0

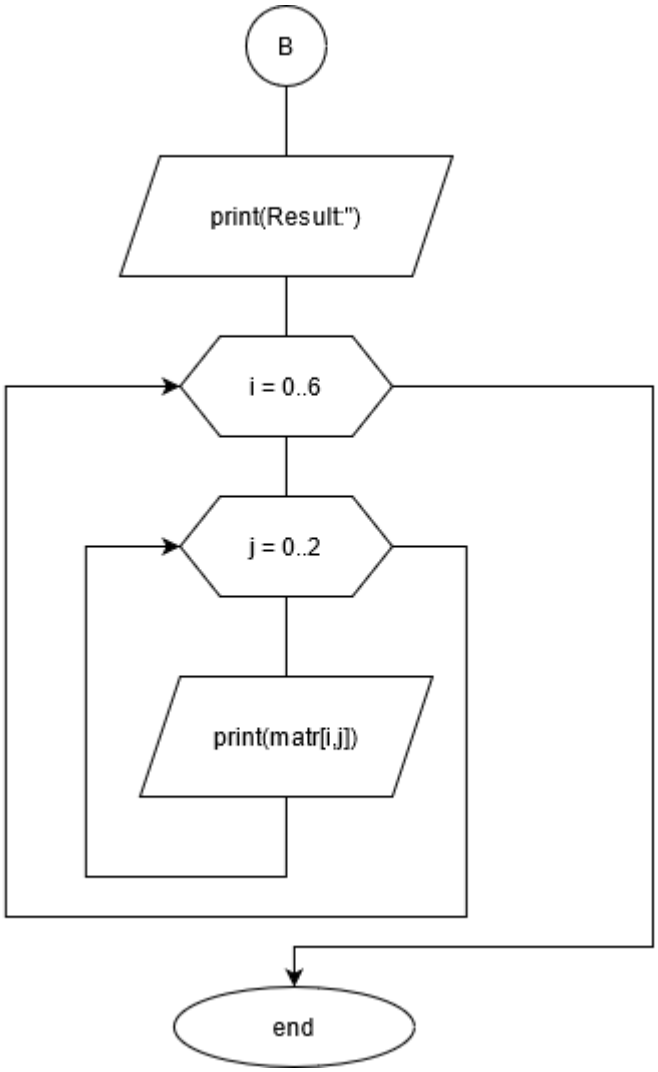


Рисунок 1 — схема алгоритма

Контрольные вопросы

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

Так как язык ассемблера работает с мнемоническими аналогами машинных команд, которые как правило адресуют массивы и матрицы посредство хранения базового адреса(адреса начала массива/матрицы) и смещения.

2. Как в ассемблере моделируются массивы?

Как последовательность элементов в памяти. Обработка как правило осуществляется с помощью адреса начала последовательности и величины смещения.

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

Регистры используются для хранения частей адреса, так как в большинстве команд ассемблера нельзя производить операции с 2 и более ячейками памяти.

4. Как в памяти компьютера размещаются элементы матриц?

Элементы матрицы размещаются в памяти последовательно, запись идет или по строкам или по столбцам.

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

При моделировании матриц она «разрезается» на строки или столбцы, которые хранятся в памяти как последовательность массивов.

Один регистр для обработки матрицы используется, как правило, если нужно обработать диагональ или 1 конкретную строку/столбец.

Два регистра используются для обработки матрицы, когда необходимо обработать все элементы или производятся вычисления связанные с обоими индексами (в принципе в некоторых случаях достаточно и одного регистра, но алгоритм становится значительно более сложным для отладки).

Вывод: в ходе данной лабораторной работы были изучены основы моделирования массивов и матриц на языке ассемблера.