

**PROJECT REPORT**

**Members:**

1. **Syed Abdul Rehman 21K-3156**
2. **Raja Ibrahim 21K-4585**

**Instructor:**

**Aamir Ali**

**Table of contents:**

1. Introduction 3
2. .DATA 4
3. Outputs 5
4. Conclusion 8
5. Appendix 9

**Introduction:**

***Matrix Calculator***

**Acknowledgments*:***

First up of all, we would like to thank Sir Aamir Ali, for his constant support in the

entire semester and in this project, he helped us in different ways, by answering us our queries

and helping us in improving our goals - achievement.

**About**:

This calculator is a matrices-based calculator in Assembly language, which has 10 functions within it. Each time user is asked to input and then output is shown usually in the form of array. Code is user friendly, he or she can solve any type of matrices problem from the 10 listed problems, but within the limit of dimensions specified.

**Tools and techniques:**

Microsoft visual studio 2019, Assembly 8086 MASM.

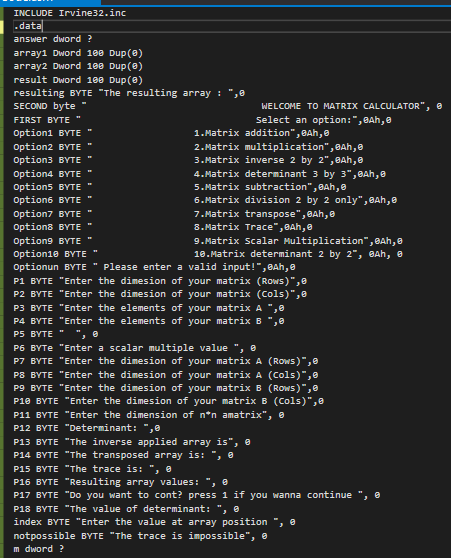
Conditional and unconditional jumps, arrays, direct indirect operands, Irvine library procedures, procedures, advanced procedure, jumps instructions, stack its operations and nested procedures, Primitive String Instructions, and much more.

**Application**:

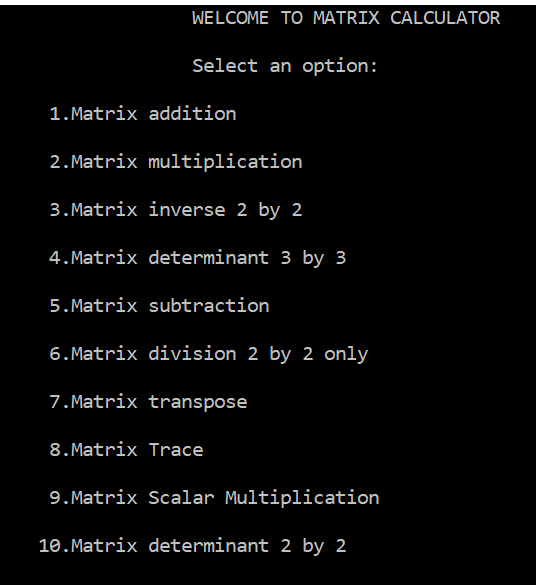
It could be used in matrix calculations.

.DATA

Data used in our code:



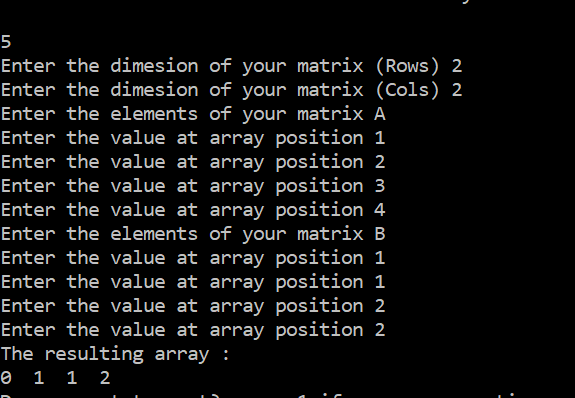
Console image once opened:



**Operations Output Screen Shots:**

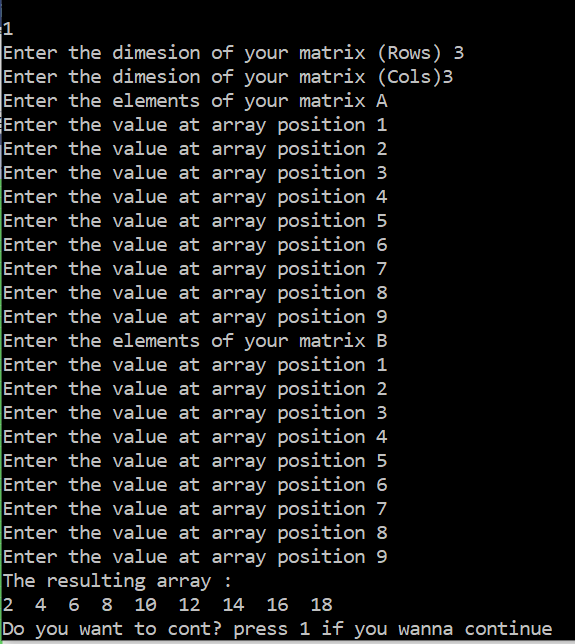
1. **Subtraction:**

Output:



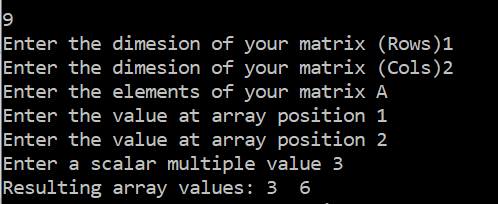
1. **Addition:**

Output:



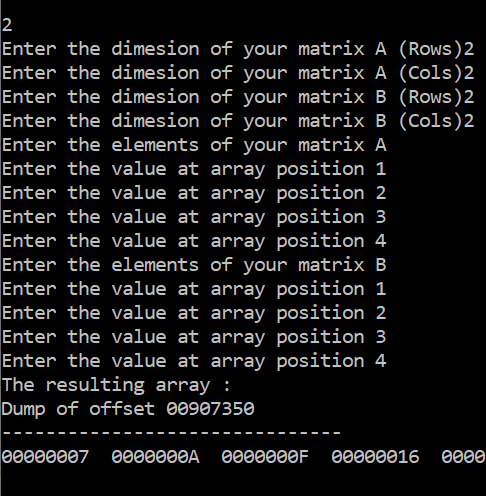
1. **Scalar Product**:

Output:



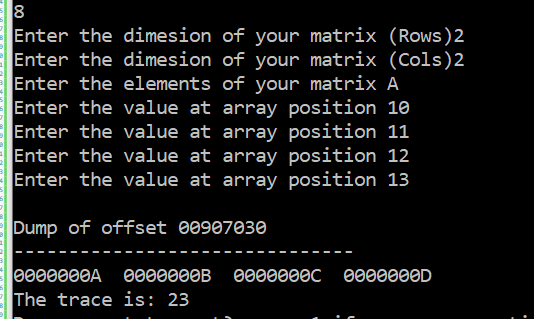
1. **Matrix Multiplication:**

Output:



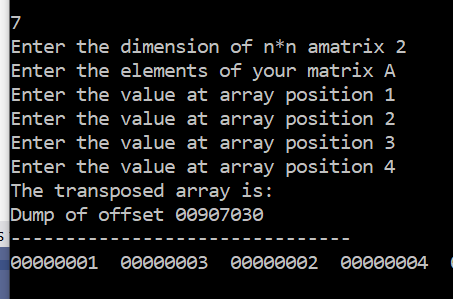
1. **Matrix trace**

Output:



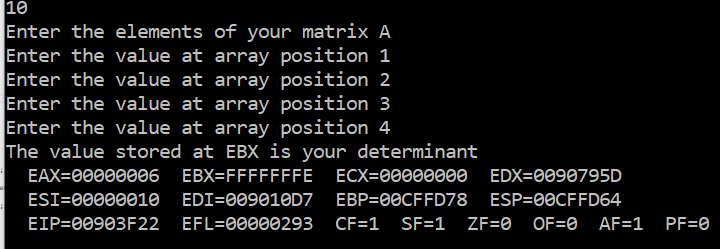
1. **Matrix Transpose**

Output:



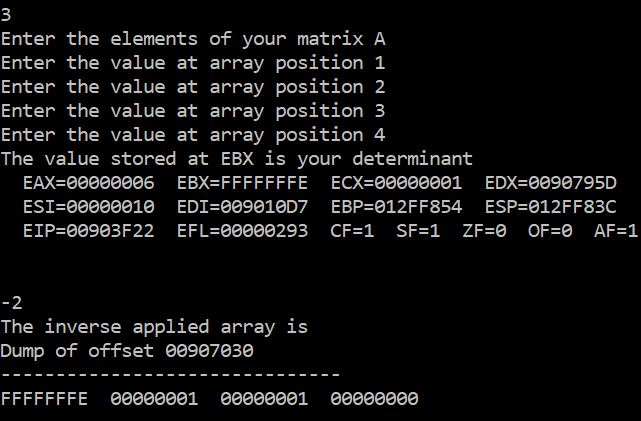
1. **Determinant 2 by 2**

Output**:**



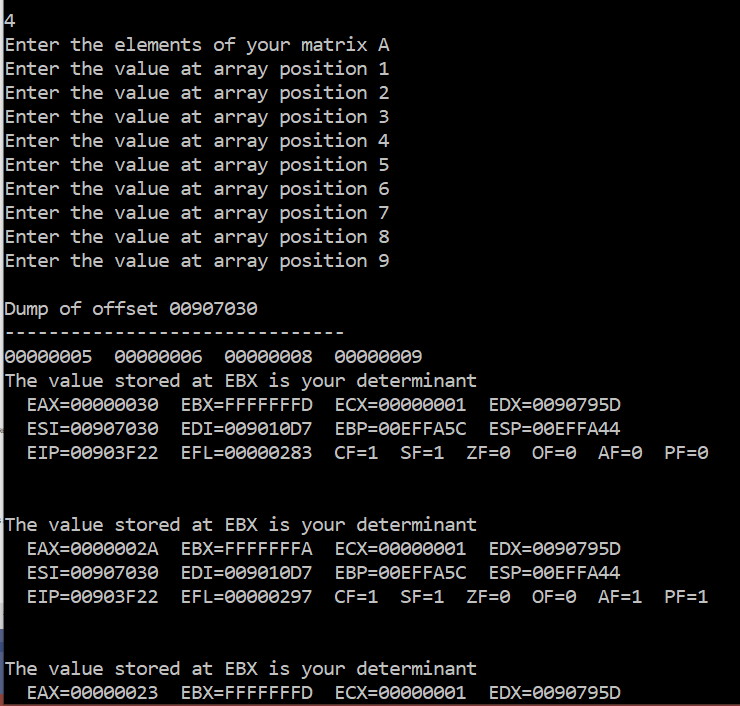
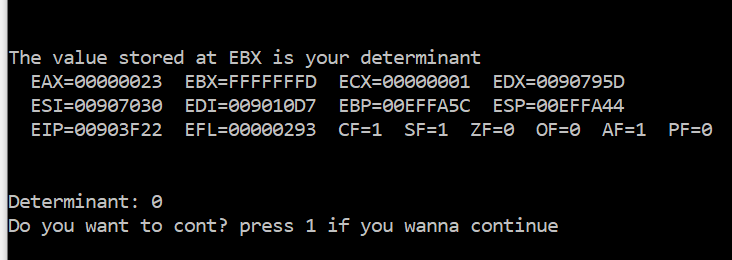
1. **Matrix inverse**

Output:



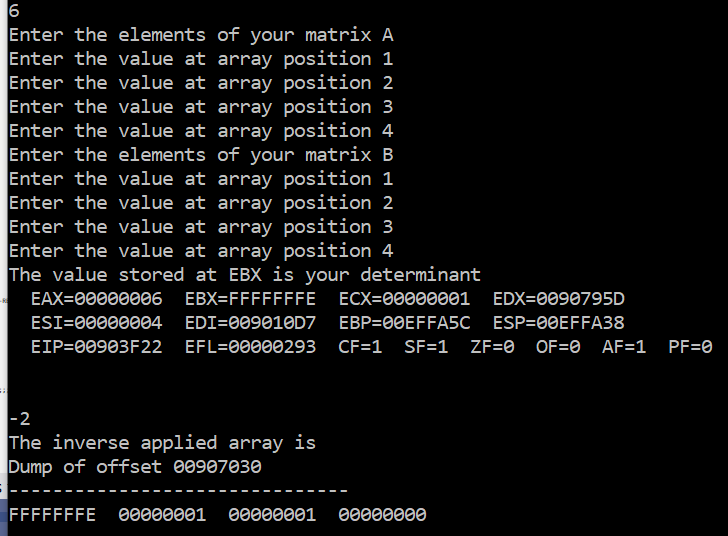
1. **Matrix determinant 3 by 3:**

Output:

1. **Divison**:

Output:



Above mentioned are the functions we used in our code.

# Conclusion:

This was our project on Matrices Calculator using Assembly 8086. It enhanced our grasp on this course and improved our coding skills using assembly language.

Thank You

**Appendix**

The code we made is attached here for clarifications:

INCLUDE Irvine32.inc

.data

answer dword ?

array1 Dword 100 Dup(0)

array2 Dword 100 Dup(0)

result Dword 100 Dup(0)

resulting BYTE "The resulting array : ",0

SECOND byte " WELCOME TO MATRIX CALCULATOR", 0

FIRST BYTE " Select an option:",0Ah,0

Option1 BYTE " 1.Matrix addition",0Ah,0

Option2 BYTE " 2.Matrix multiplication",0Ah,0

Option3 BYTE " 3.Matrix inverse 2 by 2",0Ah,0

Option4 BYTE " 4.Matrix determinant 3 by 3",0Ah,0

Option5 BYTE " 5.Matrix subtraction",0Ah,0

Option6 BYTE " 6.Matrix division 2 by 2 only",0Ah,0

Option7 BYTE " 7.Matrix transpose",0Ah,0

Option8 BYTE " 8.Matrix Trace",0Ah,0

Option9 BYTE " 9.Matrix Scalar Multiplication",0Ah,0

Option10 BYTE " 10.Matrix determinant 2 by 2", 0Ah, 0

Optionun BYTE " Please enter a valid input!",0Ah,0

P1 BYTE "Enter the dimesion of your matrix (Rows)",0

P2 BYTE "Enter the dimesion of your matrix (Cols)",0

P3 BYTE "Enter the elements of your matrix A ",0

P4 BYTE "Enter the elements of your matrix B ",0

P5 BYTE " ", 0

P6 BYTe "Enter a scalar multiple value ", 0

P7 BYTE "Enter the dimesion of your matrix A (Rows)",0

P8 BYTE "Enter the dimesion of your matrix A (Cols)",0

P9 BYTE "Enter the dimesion of your matrix B (Rows)",0

P10 BYTE "Enter the dimesion of your matrix B (Cols)",0

P11 BYTE "Enter the dimension of n\*n amatrix", 0

P12 BYTE "Determinant: ",0

P13 BYTE "The inverse applied array is", 0

P14 BYTE "The transposed array is: ", 0

P15 BYTE "The trace is: ", 0

P16 BYTE "Resulting array values: ", 0

P17 BYTE "Do you want to cont? press 1 if you wanna continue ", 0

P18 BYTE "The value of determinant: ", 0

index BYTE "Enter the value at array position ", 0

notpossible BYTE "The trace is impossible", 0

m dword ?

.code

main PROC

call mainm

exit

main ENDP

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

mainm PROC

;;;;;mWriteln "Ali"

MOV EDX, OFFSET SECOND

CALL WRITESTRING

call crlf

call crlf

mov edx,OFFSET FIRST

call writestring

call crlf

mov edx,OFFSET Option1

call writestring

call crlf

mov edx,OFFSET Option2

call writestring

call crlf

mov edx,OFFSET Option3

call writestring

call crlf

mov edx,OFFSET Option4

call writestring

call crlf

mov edx,OFFSET Option5

call writestring

call crlf

mov edx,OFFSET Option6

call writestring

call crlf

mov edx,OFFSET Option7

call writestring

call crlf

mov edx,OFFSET Option8

call writestring

call crlf

mov edx,OFFSET Option9

call writestring

call crlf

mov edx,OFFSET Option10

call writestring

call crlf

call readint

CMP eax,1

je MADD

CMP eax,2

je MM

CMP eax,3

je MI

CMP eax,4

je MDT

CMP eax,5

je MS

CMP eax,6

je MDIV

CMP eax,7

je MTRA

CMP eax,8

je MT

CMP eax,9

je SM

CMP eax, 10

JE D2\_

jne DE\_

MADD:

call MatrixAdd

jmp rt

MM:

call matrix\_Mul

jmp rt

D2\_:

call Det2

jmp rt

MI:

call MatrixInv

jmp rt

MDT:

call Matrixdet

jmp rt

MS:

call MatrixSub

jmp rt

MDIV:

call MatrixDiv

jmp rt

MTRA:

call MatrixTrans

jmp rt

MT:

call MatrixTrace

jmp rt

SM:

call Scalar

jmp rt

DE\_:

mov edx,OFFSET Optionun

call writestring

call waitmsg

call crlf

mov edx, offset P17

call writestring

call readint

cmp eax, 1

JNE rt

call mainm

rt:

ret

mainm ENDP

MatrixSub PROC

local x:DWORD,y:DWORD

mov edx, offset P1

call writestring

call readint

mov x, eax

mov edx, offset P2

call writestring

call readint

mov y, eax

mul x

mov ecx, eax

mov esi, 0

mov edx, offset P3

call writestring

call crlf

L1:

mov edx, offset index

call writestring

mov eax,esi

call writedec

mov al,'>'

call writechar

call readint

mov array1[esi\*4],eax

ADD esi,1

Loop L1

mov eax, x

mul y

mov ecx, eax

mov esi, 0

mov edx, offset P4

call writestring

call crlf

L2:

mov edx, offset index

call writestring

mov eax,esi

call writedec

mov al,'>'

call writechar

call readint

mov array2[esi\*4],eax

ADD esi,1

Loop L2

mov eax, x

mul y

mov ecx, eax

mov esi, 0

mov edx, offset resulting

call writestring

call crlf

L3:

mov eax, [array1+esi]

sub eax, [array2+esi]

mov [result+esi], eax

call writeint

mov edx, offset P5

call writestring

add esi, 4

Loop L3

call crlf

mov edx, offset P17

call writestring

call readint

cmp eax, 1

JNE end2

call mainm

end2:

ret

MatrixSub ENDP

MatrixAdd PROC

local x:DWORD,y:DWORD

mov edx, offset P1

call writestring

call readint

mov x, eax

mov edx, offset P2

call writestring

call readint

mov y, eax

mul x

mov ecx, eax

mov esi, 0

mov edx, offset P3

call writestring

call crlf

L1:

mov edx, offset index

call writestring

mov eax,esi

call writedec

mov al,'>'

call writechar

call readint

mov array1[esi\*4],eax

ADD esi,1

Loop L1

mov eax, x

mul y

mov ecx, eax

mov esi, 0

mov edx, offset P4

call writestring

call crlf

L2:

mov edx, offset index

call writestring

mov eax,esi

call writedec

mov al,'>'

call writechar

call readint

mov array2[esi\*4],eax

ADD esi,1

Loop L2

mov eax, x

mul y

mov ecx, eax

mov esi, 0

mov edx, offset resulting

call writestring

call crlf

L3:

mov eax, [array1+esi]

add eax, [array2+esi]

mov [result+esi], eax

call writedec

mov edx, offset P5

call writestring

add esi, 4

Loop L3

call crlf

mov edx, offset P17

call writestring

call readint

cmp eax, 1

JNE end2

call mainm

end2:

ret

MatrixAdd ENDP

Scalar PROC

local x:DWORD,y:DWORD

mov edx, offset P1

call writestring

call readint

mov x, eax

mov edx, offset P2

call writestring

call readint

mov y, eax

mul x

mov ecx, eax

mov esi, 0

mov edx, offset P3

call writestring

call crlf

L1:

mov edx, offset index

call writestring

mov eax,esi

call writedec

mov al,'>'

call writechar

call readint

mov array1[esi\*4],eax

ADD esi,1

Loop L1

mov edx, offset P6

call writestring

call readint

mov m, eax

mov esi, offset array1

mov edi, offset result

mov eax, x

mul y

mov ecx, eax

mov edx, offset P16

call writestring

L2:

lodsd

mul m

stosd

;mov [result+edi], eax

call writedec

mov edx, offset P5

call writestring

;add esi, 4

Loop L2

call crlf

mov edx, offset P17

call writestring

call readint

cmp eax, 1

JNE end2

call mainm

end2:

ret

Scalar ENDP

matrix\_Mul PROC

local row1:DWORD,col1:DWORD, row2:DWORD,col2:DWORD

;u:DWORD,v:DWORD

;--------------------------------GETTING DIMENSIONS OF TWO MATRICES--------------

cmp ecx, 1

JE L6

mov edx, offset P7

call writestring

call readint

mov row1,eax

mov edx, offset P8

call writestring

call readint

mov col1,eax

mov edx, offset P9

call writestring

call readint

mov row2,eax

mov edx, offset P10

call writestring

call readint

mov col2,eax

;---------------------------------------------IS MULTIPLICATION POSSIBLE

mov eax,col1

mov ebx,row2

CMP eax,ebx

JNE last

mov edx, offset P3

call writestring

call crlf

;------------------------------------------------READING MATRIX 1

mov eax,row1

mul col1

mov ecx,eax

mov esi,0

l1:

mov edx, offset index

call writestring

mov eax,esi

call writedec

mov al,'>'

call writechar

call readint

mov array1[esi\*4],eax

;call crlf

;call writedec

add esi,1

loop l1

mov edx, offset P4

call writestring

call crlf

;------------------------------------------------READING MATRIX 2

mov eax,row2

mul col2

mov ecx,eax

mov esi,0

l2:

mov edx, offset index

call writestring

mov eax,esi

call writedec

mov al,'>'

call writechar

call readint

mov array2[esi\*4],eax

;call crlf

;call writedec

add esi,1

loop l2

;print both matrices

;mov ecx,9

;mov esi,OFFSET m1

;mov ebx,4

;call dumpmem

;mov ecx,9

;mov esi,OFFSET m2

;mov ebx,4

;call dumpmem

JMP L7

L6:

mov row1, 2

mov col1, 2

mov row2, 2

mov col2, 2

L7:

;----------------------------------------------------MULTIPLICATION

mov esi,0 ;i

mov edi,0 ;j

mov ebx,0 ;k

mov ecx,row1

l3:

push ecx

mov ecx,col2

mov edi,0

l4:

push ecx

mov ecx,col1

mov eax,esi

mul col2

add eax,edi

mov answer,eax ;------------------------------ edx has index of result

mov ebx, 0

l5:

push esi

push edi

mov eax,esi

mul col1

add eax,ebx

mov esi,eax

mov eax , ebx

mul col2

add eax, edi

mov edi,eax

mov eax,array1[esi\*4]

mul array2[edi\*4]

mov edx, answer

add result[edx\*4],eax

pop edi

pop esi

inc ebx

loop l5

inc edi

pop ecx

loop l4

inc esi

pop ecx

loop l3

last:

mov edx, offset resulting

call writestring

mov eax,col1

mul row2

mov ecx,eax

mov esi,OFFSET result

mov ebx,4

call dumpmem

call crlf

mov edx, offset P17

call writestring

call readint

cmp eax, 1

JNE end2

call mainm

end2:

ret

matrix\_Mul ENDP

MatrixTrace PROC

local c\_:dword,b\_:dword

mov edx, offset P1

call writestring

call readint

mov b\_, eax

mov edx, offset P2

call writestring

call readint

mov c\_, eax

mov ebx, c\_

CMP b\_, ebx

jne not\_

mul b\_

mov ecx, eax

mov esi, 0

mov edx, offset P3

call writestring

call crlf

L1:

mov edx, offset index

call writestring

mov eax,esi

call writedec

mov al,'>'

call writechar

call readint

mov array1[esi\*4],eax

ADD esi,1

Loop L1

;mov ecx,4

;mov ebx,4

;mov esi,Offset array1

;call dumpmem

mov ebx,0

mov ecx, b\_

mov esi,0 ;i

mov edi,0 ;j

L2:

push ecx

mov ecx, c\_

mov edi,0

L3:

cmp esi, edi

jnz A

mov eax, esi

mul c\_

add eax, edi

;mov ebx, eax

add ebx, array1[eax\*4]

A:

add edi,1

loop l3

add esi,1

pop ecx

loop l2

mov eax, ebx

mov edx, offset p15

call writestring

call writeint

JMP l4

not\_:

mov edx, offset notpossible

call writestring

l4:

call crlf

mov edx, offset P17

call writestring

call readint

cmp eax, 1

JNE end2

call mainm

end2:

ret

MatrixTrace endp

MatrixTrans PROC

local t:DWORD

mov edx, offset P11

call writestring

call readint

mov t,eax

;--------------------------------------------------READING MATRIX

mov eax,t

mul t

mov ecx,eax

mov esi,0

mov edx, offset P3

call writestring

call crlf

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;mov eax, 0

t1:

mov edx, offset index

call writestring

mov eax,esi

call writedec

mov al,'>'

call writechar

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;call writedec

call readint

mov array1[esi\*4],eax

add esi,1

loop t1

;------------------------------------------------------TRANSPOSING

mov esi,0

mov ecx,t

t2:

push ecx

mov ecx,t

mov edi,0

t3:

CMP edi,esi

JG t4

push esi

push edi

mov eax ,esi

mul t

add eax,edi

mov ebx,eax ;one index

mov eax, edi

mul t

add eax,esi

mov edx,array1[eax\*4]

mov edi,eax ;two index

mov eax , array1[ebx\*4]

mov array1[ebx\*4],edx

mov array1[edi\*4],eax

pop edi

pop esi

inc edi

loop t3

t4:

inc esi

pop ecx

loop t2

mov edx, offset P14

call writestring

mov eax, t

mul eax

mov ecx, eax

mov esi,OFFSET array1

mov ebx,4

call dumpmem

call crlf

mov edx, offset P17

call writestring

call readint

cmp eax, 1

JNE end2

call mainm

end2:

ret

MatrixTrans ENDp

Det2 PROC

local state:DWORD

mov state,ecx

CMP ecx,1

JE DNI

mov ecx,4

mov esi,0

mov edx, offset P3

call writestring

call crlf

d1:

mov edx, offset index

call writestring

mov eax,esi

call writedec

mov al,'>'

call writechar

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;call writedec

call readint

mov array1[esi\*4],eax

add esi,1

loop d1

DNI:

mov eax, array1[0]

mul array1[12]

mov ebx,eax

mov eax , array1[4]

mul array1[8]

mov edx, offset P18

call writestring

sub ebx,eax

mov eax,ebx

call writeint

mov eax,state

CMP eax,1

JNE goMain

ret

goMain:

;mov eax, ebx

;mov edx, offset P12

;call writestring

;call writedec

call crlf

mov edx, offset P17

call writestring

call readint

cmp eax, 1

JNE end2

call mainm

end2:

ret

Det2 ENDP

MatrixInv PROC

CMP ecx,1

JE i3

mov ecx,4

mov esi,0

mov edx, offset P3

call writestring

call crlf

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;mov eax, 0

i1:

mov edx, offset index

call writestring

mov eax,esi

call writedec

mov al,'>'

call writechar

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;call writedec

call readint

mov array1[esi\*4],eax

add esi,1

loop i1

i3:

mov ecx,1

mov eax,array1[0]

mov edx,array1[12]

mov array1[12],eax

mov array1[0],edx

NEG array1[4]

NEG array1[8]

call Det2

mov eax,ebx

;call writeint

call crlf

mov ecx ,4

mov esi,0

i2:

mov eax,array1[esi\*4]

cdq

idiv ebx

mov array1[esi\*4],eax

inc esi

loop i2

mov edx, offset P13

call writestring

mov ecx,4

mov esi,OFFSET array1

mov ebx,4

call dumpmem

call crlf

mov edx, offset P17

call writestring

call readint

cmp eax, 1

JNE end2

call mainm

end2:

ret

MatrixInv ENDP

Matrixdet PROC

local det:DWORD

mov det,0

;----------------------------------------READING MATIRX

mov ecx,9

mov esi,0

mov edx, offset P3

call writestring

call crlf

de:

mov edx, offset index

call writestring

mov eax,esi

call writedec

mov al,'>'

call writechar

;;;;;;;;;;;;;;;;;;;;;;;;;;;call writedec

call readint

mov array2[esi\*4],eax

inc esi

loop de

mov eax,array2[16]

mov array1[0],eax

mov eax,array2[20]

mov array1[4],eax

mov eax,array2[28]

mov array1[8],eax

mov eax,array2[32]

mov array1[12],eax

mov ecx,4

mov esi,OFFSET array1

mov ebx,4

call dumpmem

mov ecx,1

call Det2

mov eax,array2[0]

mul ebx

add det,eax

mov eax,array2[12]

mov array1[0],eax

mov eax,array2[20]

mov array1[4],eax

mov eax,array2[24]

mov array1[8],eax

mov eax,array2[32]

mov array1[12],eax

mov ecx,1

call Det2

mov eax,array2[4]

mul ebx

NEG eax

add det,eax

mov eax,array2[12]

mov array1[0],eax

mov eax,array2[16]

mov array1[4],eax

mov eax,array2[24]

mov array1[8],eax

mov eax,array2[28]

mov array1[12],eax

mov ecx,1

call Det2

mov eax,array2[8]

mul ebx

add det,eax

mov eax,det

mov edx, offset P12

call writestring

call writedec

call crlf

mov edx, offset P17

call writestring

call readint

cmp eax, 1

JNE end2

call mainm

end2:

ret

Matrixdet endp

MatrixDiv PROC

;----------------------------------------READING MATRICES

mov edx, offset P3

call writestring

call crlf

mov ecx , 4

mov esi , 0

;;;;;;;;;;;;;;;;;;;;;;;;;mov eax, 0

di1:

mov edx, offset index

call writestring

mov eax,esi

call writedec

mov al,'>'

call writechar

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;call writedec

call readint

mov array1[esi\*4],eax

inc esi

loop di1

mov ecx , 4

mov esi , 0

mov edx, offset P4

call writestring

call crlf

di2:

mov edx, offset index

call writestring

mov eax,esi

call writedec

mov al,'>'

call writechar

call readint

mov array2[esi\*4],eax

inc esi

loop di2

mov ecx,1

call MatrixInv

mov ecx,1

call matrix\_Mul

call crlf

mov edx, offset P17

call writestring

call readint

cmp eax, 1

JNE end2

call mainm

end2:

ret

MatrixDiv ENDP

end main