

**Name: Rabiya Tariq Paracha**

**ID: CSC-18F-100**

**SUBJECT: ASSEMBLY LANGUAGE**

**CLASS: BSCS-4B**

**PROJECT REPORT**

**TOPIC:FINDING DETERMINENT OF 3\*3 MATRIX USING ASSEMBLY LANGUAGE**

**ABSTRACT:**

The program is about to find the determinant of matrix 3\*3. The **determinant** is useful for solving linear equations, capturing how linear transformation change area or volume, and changing variables in integrals. ... The **determinant** can be viewed as a function whose input is a square **matrix** and whose output is a number. They are useful for solving linear equations, capturing how linear transformation changes volume or area, and changing variables in integrals. In this paper we survey the known major results for computing the determinant.

**KEYWORDS:** determinant ,interrupt, variables

**INTRODUCTION:**

**Assembly language**, Type of low-level **computer programming language** consisting mostly of symbolic equivalents of a particular **computer's** machine **language**. **Computers** produced by different manufacturers have different machine **languages** and require different assemblers and **assembly languages**. Among the most common tools in electrical engineering and computer science are rectangular grids of numbers known as matrices. The numbers in a matrix can represent data, and they can also represent mathematical equations. In many time-sensitive engineering applications, multiplying matrices can give quick but good approximations of much more complicated calculations.

**FEATURES:**

Another reason that matrices are so useful in computer science is that [graphs](https://news.mit.edu/newsoffice/2012/explained-graphs-computer-science-1217.html) are. In this context, a graph is a mathematical construct consisting of nodes, usually depicted as circles, and edges, usually depicted as lines between them. Network diagrams and family trees are familiar examples of graphs, but in computer science they’re used to represent everything from [operations performed](https://news.mit.edu/newsoffice/2012/making-web-applications-more-efficient-0831.html) during the execution of a computer program to the relationships characteristic of [logistics problems](https://news.mit.edu/newsoffice/2013/algorithm-extends-artificial-intelligence-technique-1114.html).

**INTERRUPTS:**

INT 10H: This interrupt is used to execute graphics based routines.

INT 21H: This is a DOS interrupt. It is used to request different DOS function.

**PROCEDURE:**

1. The program started with the graphics code to clear screen and to give colours
2. Then the code begins with registers to load the results after completion.
3. After that iinput is taken to find determinant.
4. Then determinant formula code is used.
5. Finally the result will obtained.

**CODE:**

.MODEL SMALL

.DATA

INPUT\_ROW DB ? ; CONTAIN INPUT ROW

INPUT\_COL DB ? ; CONTAIN INPUT COLUMN

MSG1 DB "ENTER THE MATRIX ELEMENTS: $"

MSG2 DB "ROWS $"

MSG3 DB " COLUMNS $"

MSG4 DB ": $"

MSG5 DB "THE DETERMINANT OF GIVEN MATRIX IS: $"

NEW\_LINE DB 0DH,0AH, '$' ; NEW LINE CODE.

TO\_PRINT DB 20 DUP('$')

RESULT LABEL BYTE

MAX DB 150

LEN DB ?

BUFFER DB 150 DUP('?')

RES LABEL BYTE

MAX0 DB 150

LEN0 DB ?

BUFFER0 DB 150 DUP('?')

FIRST\_NUM DB ?

SECOND\_NUM DB ?

THIRD\_NUM DB ?

FOURTH\_NUM DB ?

MINUS DB 00H

FIRST\_PART DB ?

FIRST\_PART\_SIGN DB 00H

SECOND\_PART DB ?

SECOND\_PART\_SIGN DB 00H

THIRD\_PART DB ?

THIRD\_PART\_SIGN DB 00H

.CODE

MAIN PROC

MOV ah,0

MOV al,14

INT 10H

MOV ah,0bh

MOV bh,00h

MOV bl,8

INT 10h

MOV bh,1

MOV bl,1

INT 10h

MOV AX, @DATA

MOV DS, AX ; INITIALIZATION OF DATA SEGMENT.

LEA DX, MSG1 ; PRINT MSG1 TO STANDARD OUTPUT.

MOV AH, 9

INT 21H

MOV INPUT\_ROW, 03H ; STORE INPUT VALUE.

MOV INPUT\_COL, 03H ; STORE INPUT VALUE.

LEA DX, NEW\_LINE ; \n

MOV AH, 9

INT 21H

XOR AX, AX

XOR BX, BX

XOR CX, CX

XOR DX, DX

CALL INPUT\_ELEMENTS

; CALCULATE DETERMINANT

CALL THREE\_D\_DETERMINANT

MOV AX, 4C00H

INT 21H

MAIN ENDP

THREE\_D\_DETERMINANT PROC

; FIRST PART

MOV SI, 6

MOV AL, RES[SI]

MOV FIRST\_NUM, AL;

MOV SI, 7

MOV AL, RES[SI]

MOV SECOND\_NUM, AL;

MOV SI, 9

MOV AL, RES[SI]

MOV THIRD\_NUM, AL;

MOV SI, 10

MOV AL, RES[SI]

MOV FOURTH\_NUM, AL;

CALL TWO\_D\_DETERMINANT

MOV FIRST\_PART, AL

MOV AL, MINUS

MOV FIRST\_PART\_SIGN, AL

MOV SI, 2

MOV AL, RES[SI]

MOV BL, FIRST\_PART

MUL BL

MOV FIRST\_PART, AL

; SECOND PART

MOV SI, 5

MOV AL, RES[SI]

MOV FIRST\_NUM, AL;

MOV SI, 7

MOV AL, RES[SI]

MOV SECOND\_NUM, AL;

MOV SI, 8

MOV AL, RES[SI]

MOV THIRD\_NUM, AL;

MOV SI, 10

MOV AL, RES[SI]

MOV FOURTH\_NUM, AL;

CALL TWO\_D\_DETERMINANT

MOV SECOND\_PART, AL

MOV AL, MINUS

MOV SECOND\_PART\_SIGN, AL

MOV SI, 3

MOV AL, RES[SI]

MOV BL, SECOND\_PART

MUL BL

MOV SECOND\_PART, AL

; THIRD PART

MOV SI, 5

MOV AL, RES[SI]

MOV FIRST\_NUM, AL; D

MOV SI, 6

MOV AL, RES[SI]

MOV SECOND\_NUM, AL; E

MOV SI, 8

MOV AL, RES[SI]

MOV THIRD\_NUM, AL; G

MOV SI, 9

MOV AL, RES[SI]

MOV FOURTH\_NUM, AL; H

CALL TWO\_D\_DETERMINANT

MOV THIRD\_PART, AL

MOV AL, MINUS

MOV THIRD\_PART\_SIGN, AL

MOV SI, 4

MOV AL, RES[SI]

MOV BL, THIRD\_PART

MUL BL

MOV THIRD\_PART, AL

; A - B + C

MOV AL, FIRST\_PART\_SIGN

CMP AL, 00H

JE EQUAL

JMP N1

EQUAL:

MOV AL, THIRD\_PART\_SIGN

CMP AL, 00H

JE P1P2

JMP P1N2

P1P2:

MOV AL, FIRST\_PART

ADD AL, THIRD\_PART

MOV FIRST\_PART, AL

MOV FIRST\_PART\_SIGN, 00H

JMP CONTINUE

P1N2:

MOV AL, FIRST\_PART

CMP AL, THIRD\_PART

JA PS1

JMP NS1

PS1:

MOV AL, FIRST\_PART

SUB AL, THIRD\_PART

MOV FIRST\_PART, AL

MOV FIRST\_PART\_SIGN, 00H

JMP CONTINUE

NS1:

MOV AL, THIRD\_PART

SUB AL, FIRST\_PART

MOV AL, FIRST\_PART

MOV FIRST\_PART\_SIGN, 01H

JMP CONTINUE

N1:

MOV AL, THIRD\_PART\_SIGN

CMP AL, 00H

JE N1P2

JMP N1N2

N1N2:

MOV AL, FIRST\_PART

ADD AL, THIRD\_PART

MOV FIRST\_PART, AL

MOV FIRST\_PART\_SIGN, 01H

JMP CONTINUE

N1P2:

MOV AL, FIRST\_PART

CMP AL, THIRD\_PART

JA NS2

JMP PS2

PS2:

MOV AL, THIRD\_PART

SUB AL, FIRST\_PART

MOV FIRST\_PART, AL

MOV FIRST\_PART\_SIGN, 00H

JMP CONTINUE

NS2:

MOV AL, FIRST\_PART

SUB AL, THIRD\_PART

MOV FIRST\_PART, AL

MOV FIRST\_PART\_SIGN, 01H

JMP CONTINUE

CONTINUE:

MOV AL, FIRST\_PART\_SIGN

CMP AL, 00H

JE POS1

JMP NEG1

POS1:

MOV AL, SECOND\_PART\_SIGN

CMP AL, 00H

JE POS1POS2

JMP POS1NEG2

POS1POS2:

MOV AL, FIRST\_PART

CMP AL, SECOND\_PART

JA POS1POS2\_A

JMP POS1POS2\_B

POS1POS2\_A:

MOV AL, FIRST\_PART

SUB AL, SECOND\_PART

MOV FIRST\_PART, AL

MOV FIRST\_PART\_SIGN, 00H

JMP END\_D

POS1POS2\_B:

MOV AL, SECOND\_PART

SUB AL, FIRST\_PART

MOV FIRST\_PART, AL

MOV FIRST\_PART\_SIGN, 01H

JMP END\_D

POS1NEG2:

MOV AL, FIRST\_PART

ADD AL, SECOND\_PART

MOV FIRST\_PART, AL

MOV FIRST\_PART\_SIGN, 00H

JMP END\_D

NEG1:

MOV AL, SECOND\_PART\_SIGN

CMP AL, 00H

JE NEG1POS2

JMP NEG1NEG2

NEG1POS2:

MOV AL, FIRST\_PART

ADD AL, SECOND\_PART

MOV FIRST\_PART, AL

MOV FIRST\_PART\_SIGN, 01H

JMP END\_D

NEG1NEG2:

MOV AL, SECOND\_PART

CMP AL, FIRST\_PART

JA NEG1NEG2\_A

JMP NEG1NEG2\_B

NEG1NEG2\_A:

MOV AL, SECOND\_PART

SUB AL, FIRST\_PART

MOV FIRST\_PART, AL

MOV FIRST\_PART\_SIGN, 00H

JMP END\_D

NEG1NEG2\_B:

MOV AL, FIRST\_PART

SUB AL, SECOND\_PART

MOV FIRST\_PART, AL

MOV FIRST\_PART\_SIGN, 01H

JMP END\_D

END\_D:

LEA DX, NEW\_LINE

MOV AH, 9

INT 21H

LEA DX, NEW\_LINE

MOV AH, 9

INT 21H

LEA DX, NEW\_LINE

MOV AH, 9

INT 21H

LEA DX, NEW\_LINE

MOV AH, 9

INT 21H

LEA DX, NEW\_LINE

MOV AH, 9

INT 21H

LEA DX, MSG5

MOV AH, 9

INT 21H

MOV AL, FIRST\_PART\_SIGN

CMP AL, 01H

JE PRINT\_MINUS

JMP PRINT

PRINT\_MINUS:

MOV DL, '-'

MOV AH, 2

INT 21H

PRINT:

LEA SI, TO\_PRINT

XOR AX, AX

MOV AL, FIRST\_PART

CALL HEX2DEC

LEA DX, TO\_PRINT

MOV AH, 9

INT 21H

RET

THREE\_D\_DETERMINANT ENDP

HEX2DEC PROC ; DO THE MATH

XOR CX, CX

MOV BX, 10

LOOP1:

XOR DX, DX

DIV BX ; REMINDER IS DL(ACCORDING TO 8086 INSTRUCTION SET)

ADD DL, 30H ; CONVERT it to ascii

PUSH DX ; SAVE it in stack

INC CX

CMP AX, 9

JG LOOP1

ADD AL, 30H

MOV [SI], AL

LOOP2:

POP AX

INC SI

MOV [SI], AL

LOOP LOOP2

RET

HEX2DEC ENDP

TWO\_D\_DETERMINANT PROC

MOV MINUS, 00H

MOV AL, FIRST\_NUM

MOV BL, FOURTH\_NUM

MUL BL

MOV FIRST\_NUM, AL

MOV AL, SECOND\_NUM

MOV BL, THIRD\_NUM

MUL BL

MOV SECOND\_NUM, AL

CMP AL, FIRST\_NUM

JA NOT\_EQUAL

MOV AL, FIRST\_NUM

SUB AL, SECOND\_NUM

JMP END\_TWO\_D\_DETERMINANT

NOT\_EQUAL:

MOV MINUS, 01H

MOV AL, SECOND\_NUM

SUB AL, FIRST\_NUM

JMP END\_TWO\_D\_DETERMINANT

END\_TWO\_D\_DETERMINANT:

RET

TWO\_D\_DETERMINANT ENDP

INPUT\_ELEMENTS PROC

MOV CL,INPUT\_ROW

XOR BX,BX

MOV SI, 2

INPUTELEMENT:

LEA DX, MSG2 ; PRINT MSG3 TO STANDARD OUTPUT.

MOV AH, 9

INT 21H

CALL PRINTNUMBER

LEA DX, MSG3 ; PRINT MSG4 TO STANDARD OUTPUT.

MOV AH, 9

INT 21H

PUSH BX

MOV BL,BH

CALL PRINTNUMBER

POP BX

LEA DX, MSG4 ; PRINT MSG5 TO STANDARD OUTPUT.

MOV AH, 9

INT 21H

MOV AH, 0AH ; READ THE INPUT.

LEA DX, RESULT

INT 21h

PUSH CX

CALL PUSH\_NUMBER

POP CX

LEA DX, NEW\_LINE ; \n

MOV AH, 9

INT 21H

INC BH

MOV AL,INPUT\_COL

CMP BH,AL

JNE INPUTELEMENT

MOV BH,0

INC BL

LOOP INPUTELEMENT

RET

INPUT\_ELEMENTS ENDP

PUSH\_NUMBER PROC

MOV DI, 2

PUSH\_NUMBER\_LOOP:

MOV AL,RESULT[DI]

CMP AL, 0DH

JE END\_PUSH\_NUMBER

SUB AL, 30H

MOV RES[SI], AL

INC SI

INC DI

INC LEN0

JMP PUSH\_NUMBER\_LOOP

END\_PUSH\_NUMBER:

RET

PUSH\_NUMBER ENDP

PRINTNUMBER PROC

MOV AL, BL

CMP AL, 0

MOV DH, 0

JNE PRINT\_ANSWER

MOV AL, '0'

MOV AH, 0EH

INT 10H

JMP FINISH

PRINT\_ANSWER:

PUSH AX

MOV AH, 0

CMP AX, 0

JE PRINT\_FINAL

MOV DL, 10

DIV DL

INC DH

JMP PRINT\_ANSWER

PRINT\_FINAL:

POP AX

MOV AL, AH

ADD AL, 30H

MOV AH, 0EH

INT 10H

DEC DH

CMP DH,0

JNE PRINT\_FINAL

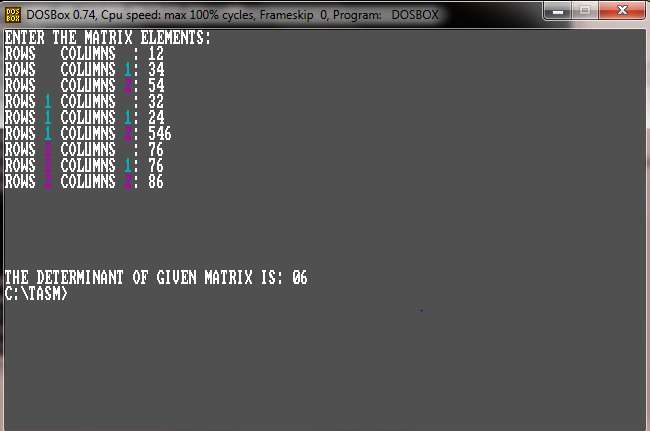
POP AX

FINISH:

RET

PRINTNUMBER ENDP

END MAIN

**OUTPUT:** 

**REFERENCES:**

<https://www.google.com/search?sxsrf=ALeKk033a1_dmbala5_oi1zvnv4yaHsywQ%3A1598825038202&ei=TiJMX5PzC5T2xgPc->

<https://news.mit.edu/2013/explained-matrices-1206>

https://www.hec.ca/en/cams/help/topics/Matrix\_determinants.pdf

https://www.sololearn.com/Discuss/103225/program-to-find-out-determinant-of-n-n-matrix-in-c-or-c