MATRIX OPERATIONS

Exp No.: 5	**Name:** S Vishakan
Date: 23-09-2020	**Reg. No:** 18 5001 196

AIM:

To write assembly language programs to perform the following matrix operations:

1. Matrix Addition
2. Matrix Subtraction

PROGRAM – 1: MATRIX ADDITION:

ALGORITHM:

1. Begin.

2. Declare the data segment.
3. Initialize data segment with matrices 1 and 2, with their dimensions and resultant matrix.
4. Close the data segment.

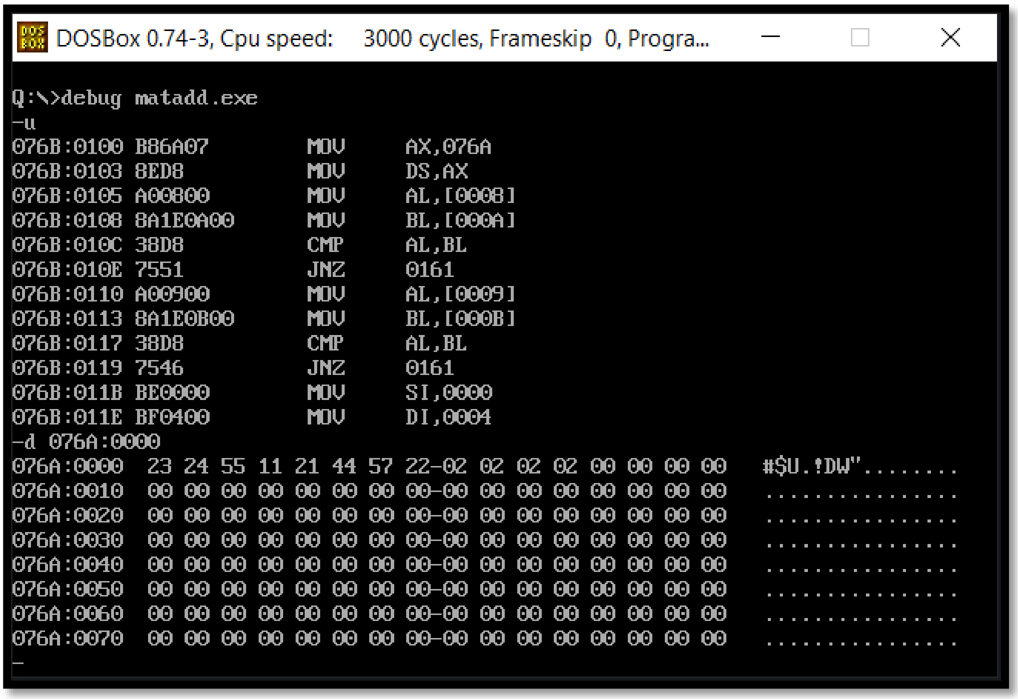
5. Declare the code segment.
6. Set a preferred offset (preferably 100)
7. Load the data segment content into AX register.
8. Transfer the contents of AX register to DS register.
9. Compare row1 and row2, if not equal then exit the program.
10. Compare col1 and col2, if not equal then exit the program.
11. Position SI at matrix1, and DI at matrix2.
12. Multiply row1 and col1 to find length len of the matrix.
13. Move the len to CL register.
14. Till CL goes to zero:
    a. Add values at SI and DI and push it into the stack.
    b. Increment SI and DI.
    c. Decrement CL.
15. Move SI to end of resultant matrix.
16. Till CL goes to zero:
    a. Pop the value from top of the stack and put it at SI.
    b. Decrement SI.
17. Introduce an interrupt for safe exit. (INT 21h)
18. Close the code segment.

19. End

PROGRAM	COMMENTS
assume cs:code, ds:data	Declare code and data segment.
data segment	Initialize data segment with values.
mat1 db 23h,24h,55h,11h	Matrix 1.
mat2 db 21h,44h,57h,22h	Matrix 2.
row1 db 02h	Row count of matrix 1.
col1 db 02h	Column count of matrix 1.
row2 db 02h	Row count of matrix 2.
col2 db 02h	Column count of matrix 2.
len db 00h	Length of matrix.
resi dw ?	Result matrix.
data ends	
code segment	Start the code segment.
org 0100h	Initialize an offset address.
start: mov ax, data	Transfer data from “data” to AX.
mov ds, ax	Transfer data from memory location AX to DS.
mov al, row1	Comparing row count of both matrices.
mov bl, row2	
cmp al, bl	
jne break	Exiting if not same.
mov al, col1	Comparing column count of both matrices.
mov bl, col2	
cmp al, bl	
jne break	Exiting if not same.
mov si, offset mat1	Set SI to point to Matrix 1’s starting index.
mov di, offset mat2	Set DI to point to Matrix 2’s starting index.
mov al, row1	
mov bl, col1	
mul bl	AL has the value of row1 * col1.
mov len, al	
mov cl, len	Finding no. of elements in the matrix.
mov ch, 00h	Clear CH.
mov ax, 0000h	Clear AX.
looper: mov al, [si]	Pushing each element-wise sum into stack
mov ah, 00h	
mov bl, [di]	
mov bh, 00h	
add ax, bx	Add the 2 elements from each matrix.
push ax	
inc si	Move to next element in matrix 1.
inc di	Move to next element in matrix 2.
dec cx	Decrement counter by 1.
jz prewrk	If addition is over, jump to “prewrk”
jmp looper	Repeat addition for all elements.

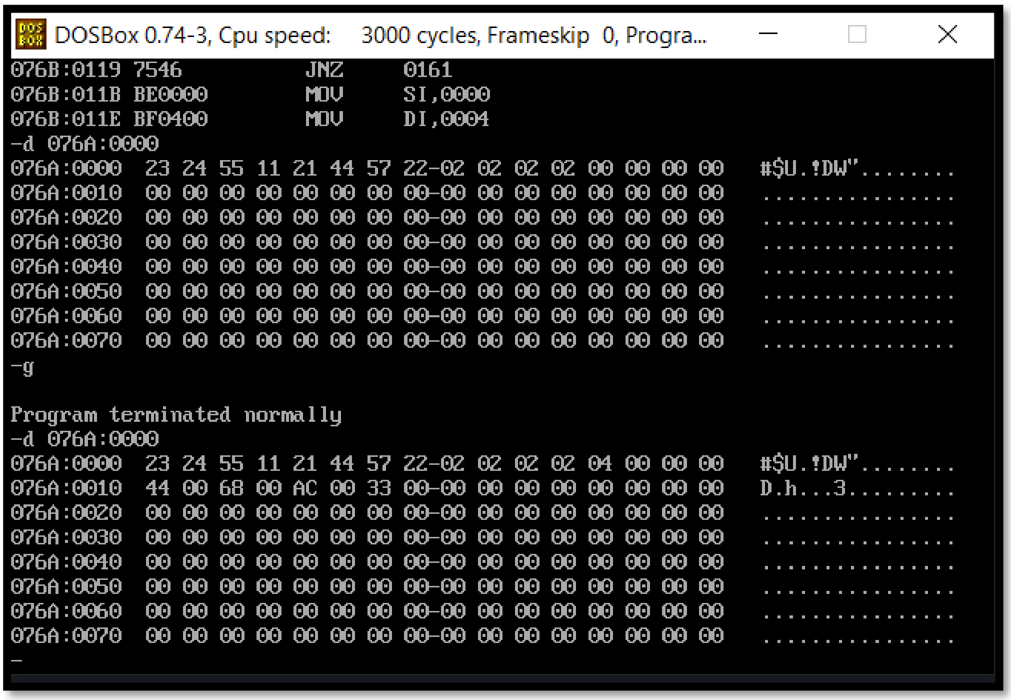
prewrk: mov si, offset resi + 0001h	Set the SI to store values in result matrix “resi” properly.
mov cl, len	Set counter to length of the matrix.
mov ch, 00h	Clear CH.
add si, cx	Set SI to point to the last location of the matrix.
retloop: pop ax	Popping each element from stack into resultant matrix.
mov [si], al	
dec si	Decrement SI.
mov [si], ah	
dec si	
dec cx	Decrement counter by 1.
jz break	Stop popping if all elements are popped (CX = 0)
jmp retloop	Pop the next element and put it in the matrix.
break: mov ah, 4ch	
int 21h	Interrupt the process with return code and exit.
code ends	
end start	

UNASSEMBLED CODE:


DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Progra...

```
Q:\>debug matadd.exe
-u
076B:0100 B86A07        MOV     AX,076A
076B:0103 8ED8          MOV     DS,AX
076B:0105 A00800        MOV     AL,[0008]
076B:0108 8A1E0A00      MOV     BL,[000A]
076B:010C 38D8          CMP     AL,BL
076B:010E 7551          JNZ     0161
076B:0110 A00900        MOV     AL,[0009]
076B:0113 8A1E0B00      MOV     BL,[000B]
076B:0117 38D8          CMP     AL,BL
076B:0119 7546          JNZ     0161
076B:011B BE0000        MOV     SI,0000
076B:011E BF0400        MOV     DI,0004
-d 076A:0000
076A:0000  23 24 55 11 21 44 57 22-02 02 02 02 00 00 00 00   #$U.!DW"........
076A:0010  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0020  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0030  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0040  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0050  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0060  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0070  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
-
```

SAMPLE I/O SNAPSHOT:


DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Progra...

```
076B:0119 7546          JNZ     0161
076B:011B BE0000        MOV     SI,0000
076B:011E BF0400        MOV     DI,0004
-d 076A:0000
076A:0000  23 24 55 11 21 44 57 22-02 02 02 02 00 00 00 00   #$U.!DW"........
076A:0010  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0020  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0030  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0040  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0050  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0060  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0070  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
-g

Program terminated normally
-d 076A:0000
076A:0000  23 24 55 11 21 44 57 22-02 02 02 04 00 00 00 00   #$U.!DW"........
076A:0010  44 00 68 00 AC 00 33 00-00 00 00 00 00 00 00 00   D.h...3.........
076A:0020  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0030  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0040  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0050  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0060  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0070  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
-
```

PROGRAM – 2: MATRIX SUBTRACTION:

ALGORITHM:

1. Begin.

2. Declare the data segment.
3. Initialize data segment with matrices 1 and 2, with their dimensions and resultant matrix.
4. Close the data segment.

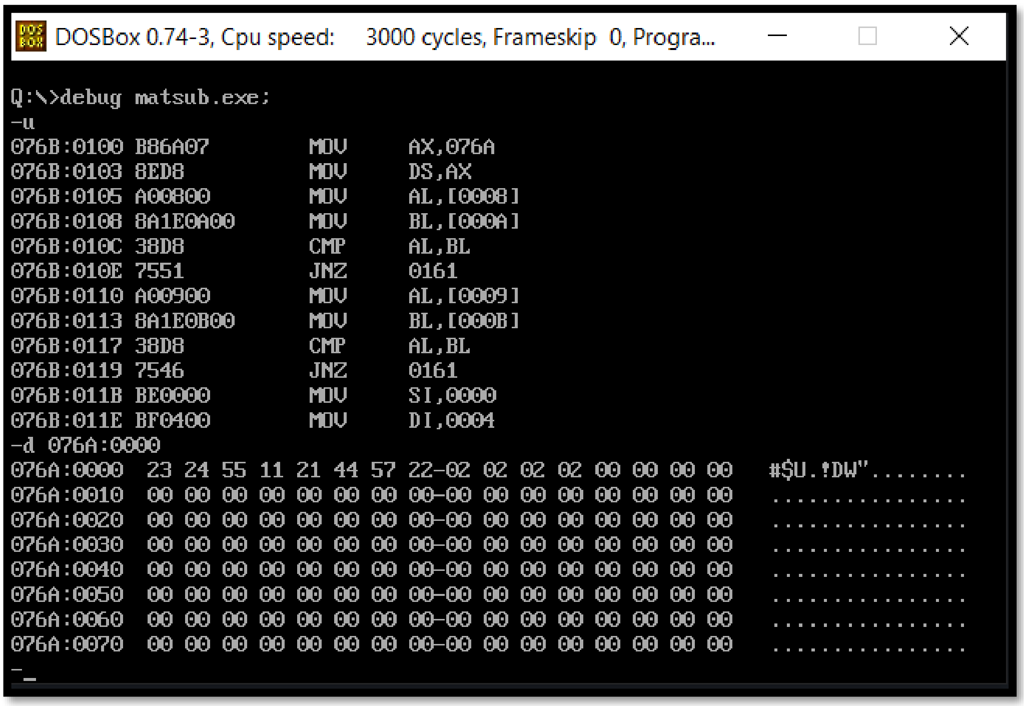
5. Declare the code segment.
6. Set a preferred offset (preferably 100)
7. Load the data segment content into AX register.
8. Transfer the contents of AX register to DS register.
9. Compare row1 and row2, if not equal then exit the program
10. Compare col1 and col2, if not equal then exit the program
11. Position SI at matrix1, and DI at matrix2.
12. Multiply row1 and col1 to find length len of the matrix.
13. Move the len to CL register.
14. Till CL goes to zero:
    a. Subtract values at SI and DI and push it into the stack.
    b. Increment SI and DI.
    c. Decrement CL.
15. Move SI to end of resultant matrix.
16. Till CL goes to zero:
    a. Pop the value from top of the stack and put it at SI.
    b. Decrement SI.
17. Introduce an interrupt for safe exit. (INT 21h)
18. Close the code segment.

19. End

PROGRAM	COMMENTS
assume cs:code, ds:data	Declare code and data segment.
data segment	Initialize data segment with values.
mat1 db 23h,24h,55h,11h	Matrix 1.
mat2 db 21h,44h,57h,22h	Matrix 2.
row1 db 02h	Row count of matrix 1.
col1 db 02h	Column count of matrix 1.
row2 db 02h	Row count of matrix 2.
col2 db 02h	Column count of matrix 2.
len db 00h	Length of matrix.
resi dw ?	Result matrix.
data ends	
code segment	Start the code segment.
org 0100h	Initialize an offset address.
start: mov ax, data	Transfer data from “data” to AX.
mov ds, ax	Transfer data from memory location AX to DS.
mov al, row1	Comparing row count of both matrices.
mov bl, row2	
cmp al, bl	
jne break	Exiting if not same.
mov al, col1	Comparing column count of both matrices.
mov bl, col2	
cmp al, bl	
jne break	Exiting if not same.
mov si, offset mat1	Set SI to point to Matrix 1’s starting index.
mov di, offset mat2	Set DI to point to Matrix 2’s starting index.
mov al, row1	
mov bl, col1	
mul bl	AL has the value of row1 * col1.
mov len, al	
mov cl, len	Finding no. of elements in the matrix.
mov ch, 00h	Clear CH.
mov ax, 0000h	Clear AX.
looper: mov al, [si]	Pushing each element-wise sum into stack
mov ah, 00h	
mov bl, [di]	
mov bh, 00h	
sub ax, bx	Subtract the 2 elements from each matrix.
push ax	
inc si	Move to next element in matrix 2.
inc di	Move to next element in matrix 1.
dec cx	Decrement counter by 1.
jz prewrk	If addition is over, jump to “prewrk”
jmp looper	Repeat addition for all elements.

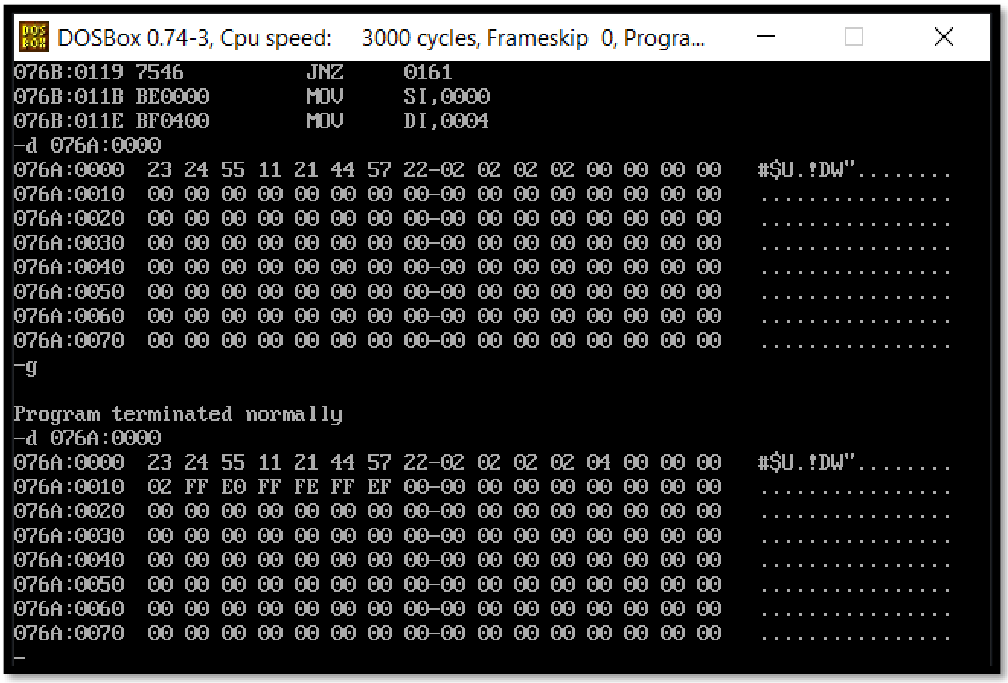
prewrk: mov si, offset resi + 0001h	Set the SI to store values in result matrix “resi” properly.
mov cl, len	Set counter to length of the matrix.
mov ch, 00h	Clear CH.
add si, cx	Set SI to point to the last location of the matrix.
add si, cx	
retloop: pop ax	Popping each element from stack into resultant matrix.
mov [si], al	
dec si	Decrement SI.
mov [si], ah	
dec si	
dec cx	Decrement counter by 1.
jz break	Stop popping if all elements are popped (CX = 0)
jmp retloop	Pop the next element and put it in the matrix.
break: mov ah, 4ch	
int 21h	Interrupt the process with return code and exit.
code ends	
end start	

UNASSEMBLED CODE:


DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Progra...

```
Q:\>debug matsub.exe;
-u
076B:0100 B86A07        MOV     AX,076A
076B:0103 8ED8          MOV     DS,AX
076B:0105 A00800        MOV     AL,[0008]
076B:0108 8A1E0A00      MOV     BL,[000A]
076B:010C 38D8          CMP     AL,BL
076B:010E 7551          JNZ     0161
076B:0110 A00900        MOV     AL,[0009]
076B:0113 8A1E0B00      MOV     BL,[000B]
076B:0117 38D8          CMP     AL,BL
076B:0119 7546          JNZ     0161
076B:011B BE0000        MOV     SI,0000
076B:011E BF0400        MOV     DI,0004
-d 076A:0000
076A:0000  23 24 55 11 21 44 57 22-02 02 02 02 00 00 00 00   #$U.!DW"........
076A:0010  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0020  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0030  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0040  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0050  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0060  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0070  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
-_
```

SAMPLE I/O SNAPSHOT:


DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Progra...

```
076B:0119 7546          JNZ     0161
076B:011B BE0000        MOV     SI,0000
076B:011E BF0400        MOV     DI,0004
-d 076A:0000
076A:0000  23 24 55 11 21 44 57 22-02 02 02 02 00 00 00 00   #$U.!DW"........
076A:0010  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0020  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0030  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0040  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0050  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0060  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0070  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
-g

Program terminated normally
-d 076A:0000
076A:0000  23 24 55 11 21 44 57 22-02 02 02 04 00 00 00 00   #$U.!DW"........
076A:0010  02 FF E0 FF FE FF EF 00-00 00 00 00 00 00 00 00   ................
076A:0020  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0030  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0040  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0050  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0060  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0070  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
-
```

RESULT:

The assembly level programs were written to perform the above specified matrix operations and the result was verified.