Matrix Operations

Expt No:	5	**Name:**	Mahesh Bharadwaj K
Date :	01/10/2020	**Reg No:**	185001089

Aim:

To write and execute 8086 programs for Matrix Operations like addition and subtraction.

Procedure:

- Mount masm folder to a drive on DOSBOX.
- Navigate to mounted drive using ‘dir’ .
- Save 8086 program with the extension **‘.asm’** in the same folder using the command **‘edit’**.
- Assemble the **.asm** file using the command **‘masm filename.asm’**.
- Link the assmebled **.obj** file using the command **‘link filename.obj’**.
- Debug the executable file **.exe** with the **‘debug filename.exe’** command.

   i. **U:** To view the un-assembled code.

   ii. **D:** Used as ‘D segment:offset’ to see the content of memory locations starting from segment:offset address.

   iii. **E:** To change the values in memory.

   iv. **G:** Execute the program using command.

   v. **Q** exits from the debug session.

Algorithm:

1. Matrix Addition

∗ The matrices are stored in mat1 and mat2 in row major format.

∗ Move the data segment address to the AX register and then move it to the DS register.

∗ Check if both matrices have same row size, if not terminate.

∗ Check if both matrices have same column size, if not terminate.

∗ Multiply row and column size and store in CX register.

∗ Load Effective Address of matrix 1 into SI using LEA.

∗ Load Effective Address of matrix 2 into DI using LEA.

∗ Load Effective Address of result matrix into BX using LEA.

∗ BEGIN LOOP

  – Move value at [SI] into AL using MOV.
  – Add value at [DI] to AL using ADD.
  – Store the value at AL into [BX].
  – Increment SI, DI & BX.
  – Decrement CX.
  – IF CX is 0, END LOOP

2. Matrix Subtraction

∗ The matrices are stored in mat1 and mat2 in row major format.

∗ Move the data segment address to the AX register and then move it to the DS register.

∗ Check if both matrices have same row size, if not terminate.

∗ Check if both matrices have same column size, if not terminate.

∗ Multiply row and column size and store in CX register.

∗ Load Effective Address of matrix 1 into SI using LEA.

∗ Load Effective Address of matrix 2 into DI using LEA.

∗ Load Effective Address of result matrix into BX using LEA.

∗ BEGIN LOOP

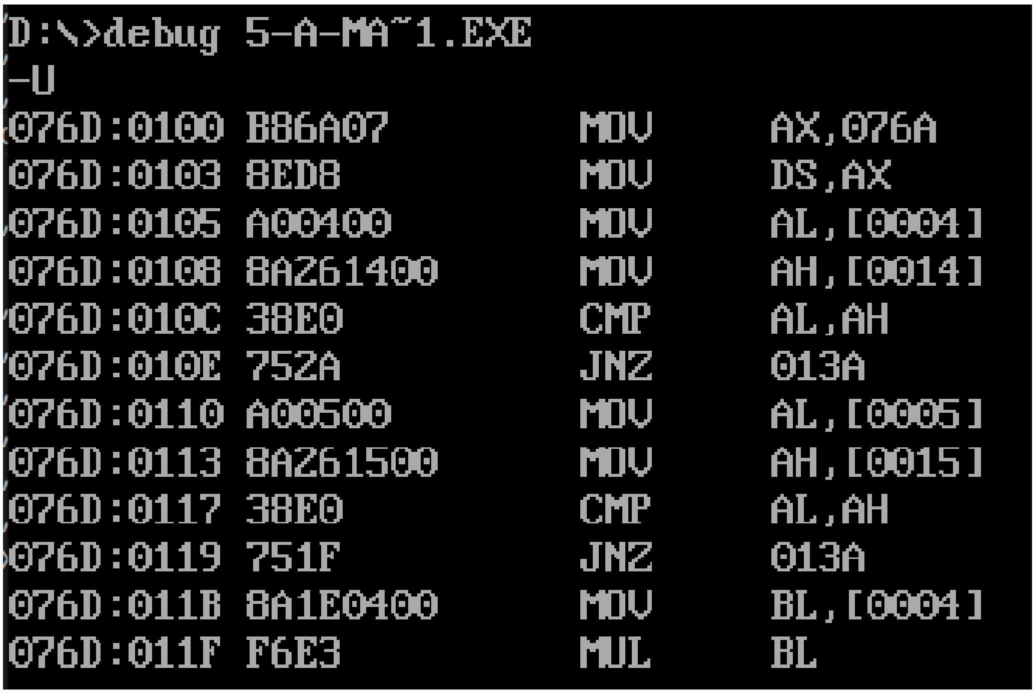
  – Move value at [SI] into AL using MOV.
  – Subtract value AL from [DI] using SUB.
  – Store the value at AL into [BX].
  – Increment SI, DI & BX.
  – Decrement CX.
  – IF CX is 0, END LOOP

1.Matrix Addition

Program:

Program	**Comments**
start: MOV AX,data	Move data segment address contents to AX register
MOV ds,AX	Move data in AX register to DS register
MOV AL, row1	Load row size of matrix 1
MOV AH, row2	Load row size of matrix 2
CMP AL, AH	Compare row sizes
JNZ **stop**	Rows are unequal, terminate
MOV AL, col1	Load col size of matrix 1
MOV AH, col2	load col size of matrix 2
CMP AL, AH	Compare column sizes
JNZ **stop**	Columns are unequal, terminate
MOV BL, row1	load row size of matrix 1
MUL BL	size of matrix is row size(in BL) * col size(in AL)
MOV CX, AX	Storing size into CX for LOOP
LEA SI, mat1	Load effective address of matrix 1
LEA DI, mat2	Load effective address of matrix 2
LEA BX, res_mat	Load effective address of result matrix
here: MOV AL, [SI]	Load operand 1 into AL
ADD AL, [DI]	Add operand 2(in [DI]) to AL
MOV [BX], AL	Store result in result matrix
INC BX	
INC SI	
INC DI	
LOOP **here**	loop till CX becomes 0
stop: MOV ah,4ch	
INT 21h	Request interrupt routine

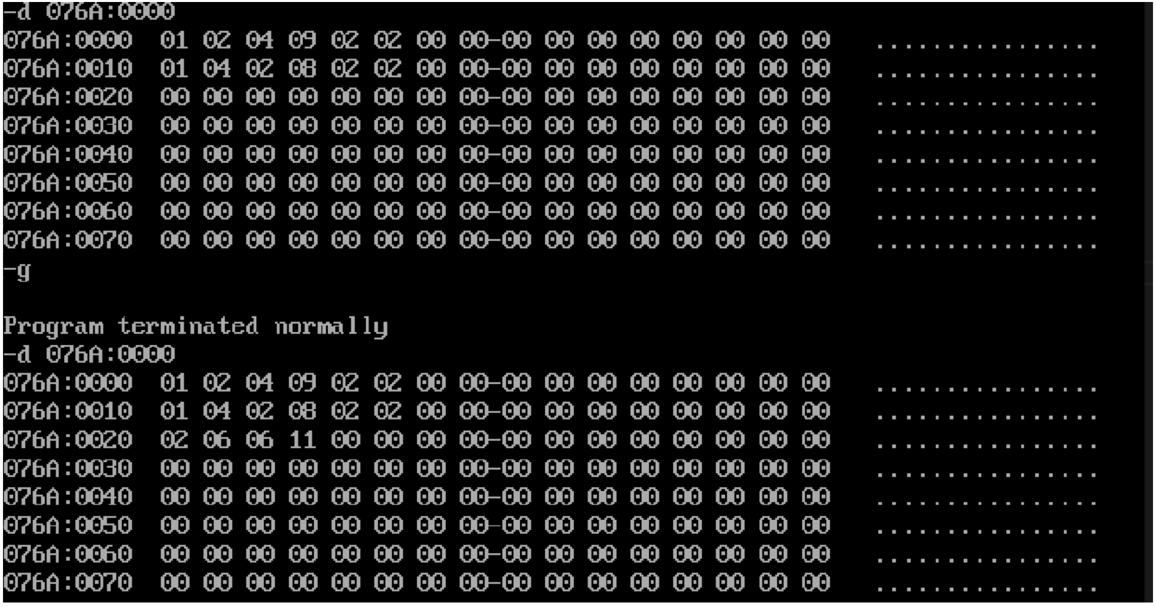
Unassembled Code:



```
D:\>debug 5-A-MA~1.EXE
-U
076D:0100 B86A07        MOV     AX,076A
076D:0103 8ED8          MOV     DS,AX
076D:0105 A00400        MOV     AL,[0004]
076D:0108 8A261400      MOV     AH,[0014]
076D:010C 38E0          CMP     AL,AH
076D:010E 752A          JNZ     013A
076D:0110 A00500        MOV     AL,[0005]
076D:0113 8A261500      MOV     AH,[0015]
076D:0117 38E0          CMP     AL,AH
076D:0119 751F          JNZ     013A
076D:011B 8A1E0400      MOV     BL,[0004]
076D:011F F6E3          MUL     BL
```

Input and Output:

Figure 1: **Input:** matrix_1 = {01h, 02h, 04h, 09h} & matrix_2 = {01h, 04h, 02h, 08h} **Output:** result_matrix = {02h, 06h, 06h, 11h}



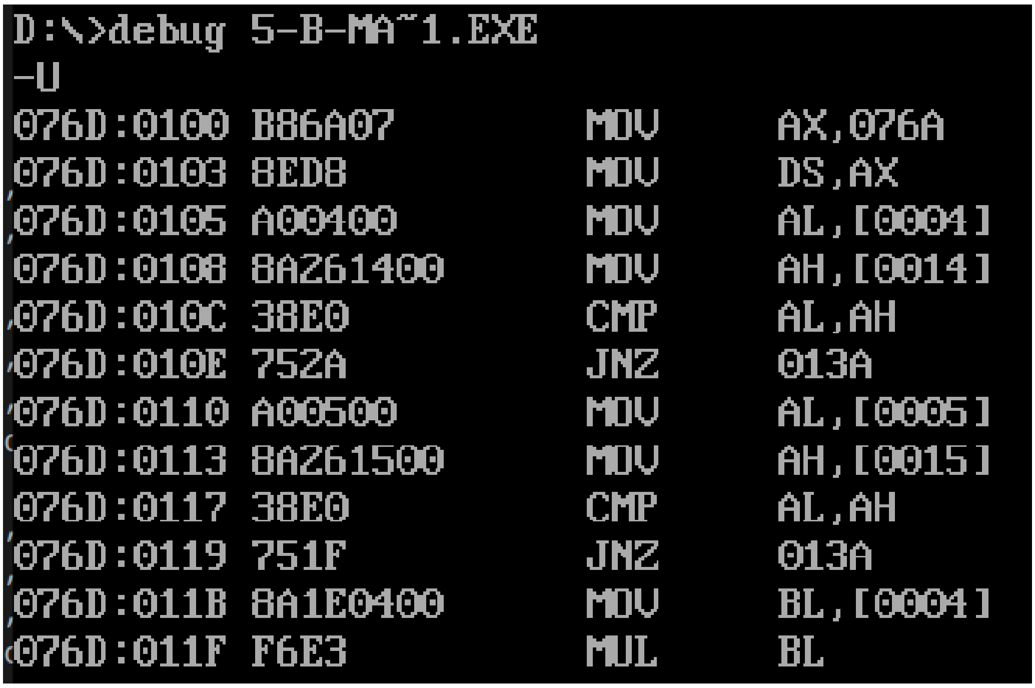
```
-d 076A:0000
076A:0000  01 02 04 09 02 02 00 00-00 00 00 00 00 00 00 00   ................
076A:0010  01 04 02 08 02 02 00 00-00 00 00 00 00 00 00 00   ................
076A:0020  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0030  00 00 00 00 00 00 00 00-00 00 00 00 00 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  01 02 04 09 02 02 00 00-00 00 00 00 00 00 00 00   ................
076A:0010  01 04 02 08 02 02 00 00-00 00 00 00 00 00 00 00   ................
076A:0020  02 06 06 11 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   ................
```

2.Matrix Subtraction

Program:

	Program	Comments
start:	MOV AX,data	Move data segment address contents to AX register
	MOV ds,AX	Move data in AX register to DS register
	MOV AL, row1	Load row size of matrix 1
	MOV AH, row2	Load row size of matrix 2
	CMP AL, AH	Compare row sizes
	JNZ **stop**	Rows are unequal, terminate
	MOV AL, col1	Load col size of matrix 1
	MOV AH, col2	load col size of matrix 2
	CMP AL, AH	Compare column sizes
	JNZ **stop**	Columns are unequal, terminate
	MOV BL, row1	load row size of matrix 1
	MUL BL	size of matrix is row size(in BL) * col size(in AL)
	MOV CX, AX	Storing size into CX for LOOP
	LEA SI, mat1	Load effective address of matrix 1
	LEA DI, mat2	Load effective address of matrix 2
	LEA BX, res_mat	Load effective address of result matrix
here:	MOV AL, [SI]	Load operand 1 into AL
	SUB AL, [DI]	Subtract operand 2(in [DI]) from AL
	MOV [BX], AL	Store result in result matrix
	INC BX	
	INC SI	
	INC DI	
	LOOP **here**	loop till CX becomes 0
stop:	MOV ah,4ch	
	INT 21h	Request interrupt routine

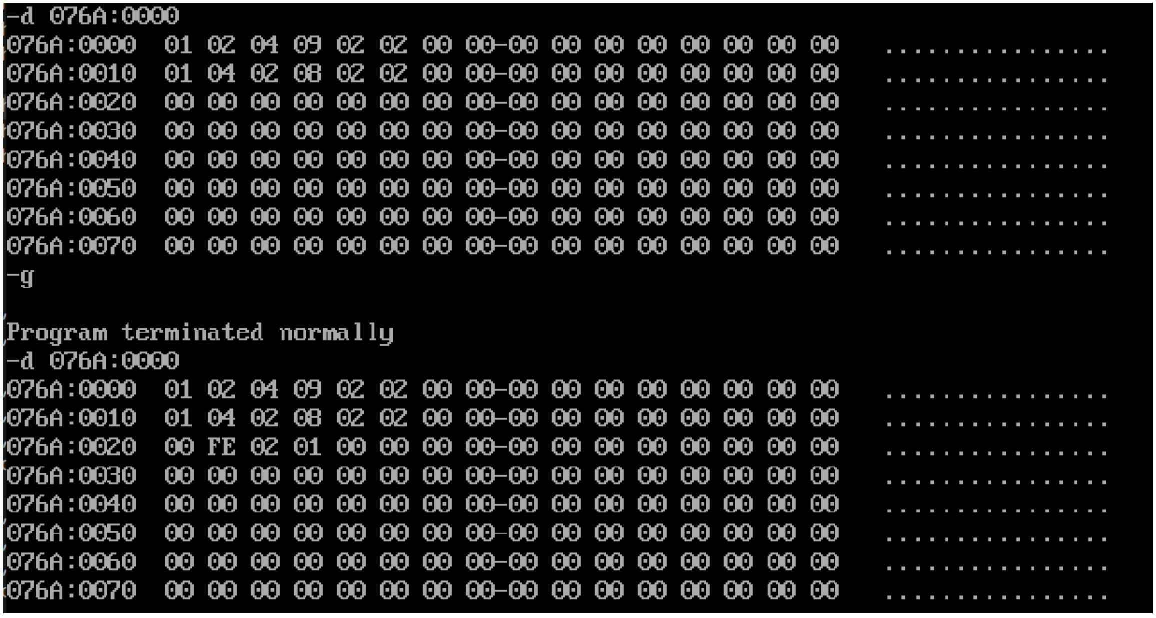
Unassembled Code:



```
D:\>debug 5-B-MA~1.EXE
-U
076D:0100 B86A07        MOV     AX,076A
076D:0103 8ED8          MOV     DS,AX
076D:0105 A00400        MOV     AL,[0004]
076D:0108 8A261400      MOV     AH,[0014]
076D:010C 38E0          CMP     AL,AH
076D:010E 752A          JNZ     013A
076D:0110 A00500        MOV     AL,[0005]
076D:0113 8A261500      MOV     AH,[0015]
076D:0117 38E0          CMP     AL,AH
076D:0119 751F          JNZ     013A
076D:011B 8A1E0400      MOV     BL,[0004]
076D:011F F6E3          MUL     BL
```

Input and Output:

Figure 2: **Input:** matrix_1 = {01h, 02h, 04h, 09h} & matrix_2 = {01h, 04h, 02h, 08h}
Output: result_matrix = {00h, FEh, 02h, 01h}



```
-d 076A:0000
076A:0000  01 02 04 09 02 02 00 00-00 00 00 00 00 00 00 00   ................
076A:0010  01 04 02 08 02 02 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  01 02 04 09 02 02 00 00-00 00 00 00 00 00 00 00   ................
076A:0010  01 04 02 08 02 02 00 00-00 00 00 00 00 00 00 00   ................
076A:0020  00 FE 02 01 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:

8086 ASL programs for Matrix Operations like addition and subtraction have been executed successfully using MS - DOSBox.