Aravind Kannan Rathinasabapathi
195001022 Date: 13-08-2021
Ex. No. 5

# Exercise 5 – Matrix operations
## 5A – Matrix Addition

## Aim:

To perform matrix addition.

## Procedure for executing MASM:

1. Mount the local folder in the DOS-BOX using a temp disk name: `mount <disk-name> <folder-location>`
2. Change directory into the mounted disk: `<disk-name>: `
3. Assemble the instructions: `masm <file-name>.asm`
4. Link the object file(s) to produce an executable file(.exe): `link <file-name>.obj; ` Note that removal of semi-colon will make linking process interactive.
5. Debug the executable file to read the memory map and execute the program: `debug <file-name>.exe`. After entering debug mode,
    a. `d <segment:offset> ` - dump(read) memory map from the given location
    b. `e <segment:offset> ` - edit memory values from the given location. Use ‘White space’ to continue editing and ‘new line’ to exit editing.
    c. `u ` - unassemble code (with or without <segment:offset>)
    d. `g ` - execute the program
    e. `? ` - display command list
    f. `q` - quit the debugger

## Algorithm:

1. Initialise data and extra segment using their respective registers.
2. Load the base address of data segment(ds) and extra segment(es) using an intermediate accumulator register(ax) as direct memory transfer is not allowed in 8086.
3. Check for equality in number of rows and columns, else, terminate.
4. Calculate the length of the matrix, row * col, to run the loop, iter and store it in CX.
5. Load effective address of mat1, mat2 into SI, DI. Move the offset value of res into BX.
6. Move content at SI to AL, add it with content at DI. Move the result back to res using BX.
7. Increment SI, DI, BX.
8. Decrement CX and loop over the body, 6 and 7, until CX != 0.
9. Terminate the program.

Aravind Kannan Rathinasabapathi Date: 13-08-2021
195001022 Ex. No. 5

## Program:

Program

```
; 5a: Program to add 2 matrices
assume cs: code, ds: data
data segment
    row1 db 03H
    col1 db 02H
    row2 db 03H
    col2 db 02H
    org 10H
    mat1 db 00H, 01H, 03H, 05H, 07H, 09H
    org 20H
    mat2 db 02H, 04H, 06H, 08H, 0aH, 0cH
    org 30H
    res db ?
data ends
code segment
start:  mov ax, data
        mov ds, ax
        ; Check row1 == row2
        mov al, row1
        cmp al, row2
        jne term
        ; Check col1 == col2
        mov bl, col1
        cmp bl, col2
        jne term
        ; length of row major
        mul bl
        mov cx, ax
        ; ptr to operands and result
        lea si, mat1
        lea di, mat2
        mov bx, offset res
iter:   mov al, [si]
        add al, [di]
        mov [bx], al
        inc si
        inc di
        inc bx
        loop iter
term:   mov ah, 4cH
        int 21H
code ends
end start
```

Comment

Comment after ';'

Map CS to code segment, DS to data segment

Initialise data segment and extra segment
db = define a byte
Initialise row1, col1, row2, col2
Initialise mat1, mat2
Define res

Initialise code segment
Move the starting address of data segment in ax, then move ax to ds.

Load AL with row1, compare it against row2 to set flag registers. Jump to terminate, if zero flag is not set.

Load BL with col1, compare it against col2 to set flag registers. Jump to terminate, if zero flag is not set.

Calculate the length of the matrix, row * col, to run the loop, iter and store it in CX.

Load effective address of mat1, mat2 into SI, DI. Move the offset value of res into BX.

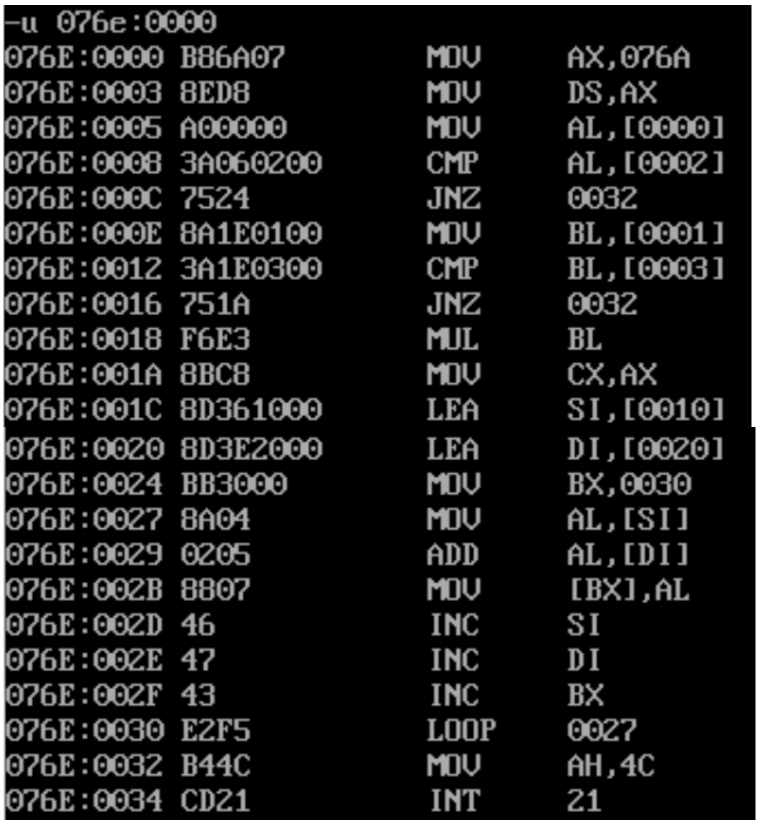
Move content at SI to AL, add it with content at DI. Move the result back to res using BX. Increment SI, DI and BX. Decrement CX and loop until CX != 0

Set ah = 4cH
Call interrupt routine 21H for DOS, which terminates if ah = 4cH

Aravind Kannan Rathinasabapathi
195001022

Date: 13-08-2021
Ex. No. 5

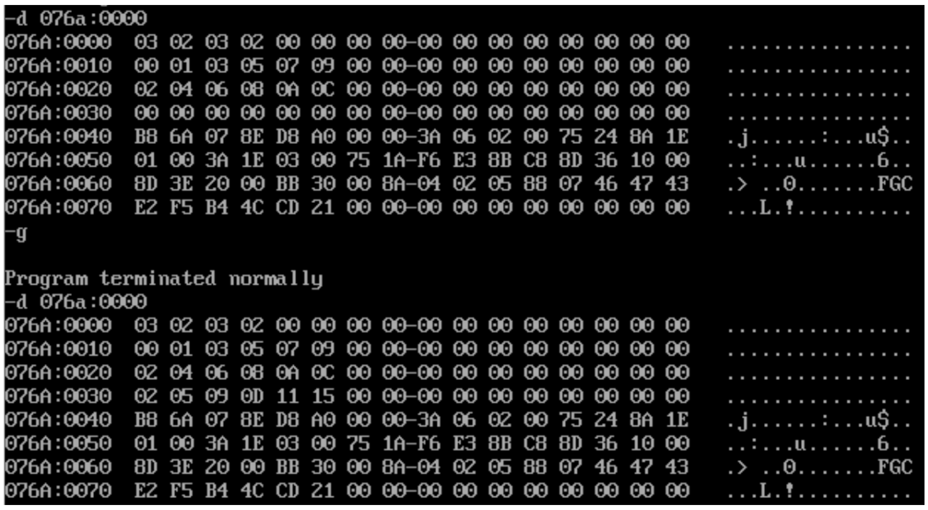
## Unassembled code:


```
-u 076e:0000
076E:0000 B86A07        MOV     AX,076A
076E:0003 8ED8          MOV     DS,AX
076E:0005 A00000        MOV     AL,[0000]
076E:0008 3A060200      CMP     AL,[0002]
076E:000C 7524          JNZ     0032
076E:000E 8A1E0100      MOV     BL,[0001]
076E:0012 3A1E0300      CMP     BL,[0003]
076E:0016 751A          JNZ     0032
076E:0018 F6E3          MUL     BL
076E:001A 8BC8          MOV     CX,AX
076E:001C 8D361000      LEA     SI,[0010]
076E:0020 8D3E2000      LEA     DI,[0020]
076E:0024 BB3000        MOV     BX,0030
076E:0027 8A04          MOV     AL,[SI]
076E:0029 0205          ADD     AL,[DI]
076E:002B 8807          MOV     [BX],AL
076E:002D 46            INC     SI
076E:002E 47            INC     DI
076E:002F 43            INC     BX
076E:0030 E2F5          LOOP    0027
076E:0032 B44C          MOV     AH,4C
076E:0034 CD21          INT     21
```

## Snapshot of sample input and output:

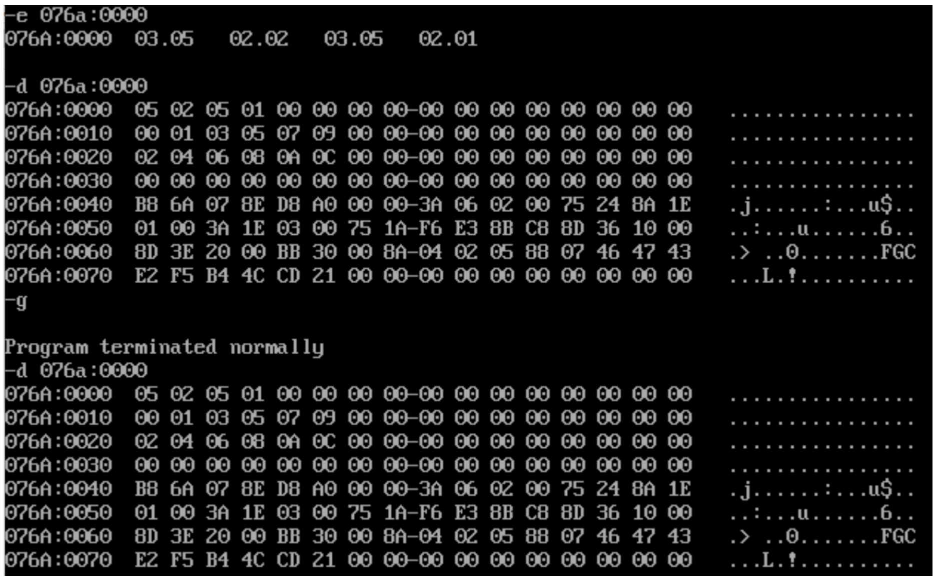
Case i: Compatible matrices

row1 = 03H col1 = 02H mat1 = [[00H, 01H], [03H, 05H], [07H, 09H]]
row2 = 03H col2 = 02H mat2 = [[02H, 04H], [06H, 08H], [0aH, 0cH]]
mat2 = [[02H, 05H], [09H, 0dH], [11H, 15H]]


```
-d 076a:0000
076A:0000  03 02 03 02 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0010  00 01 03 05 07 09 00 00-00 00 00 00 00 00 00 00   ................
076A:0020  02 04 06 08 0A 0C 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  B8 6A 07 8E D8 A0 00 00-3A 06 02 00 75 24 8A 1E   .j......:...u$..
076A:0050  01 00 3A 1E 03 00 75 1A-F6 E3 8B C8 8D 36 10 00   ..:...u......6..
076A:0060  8D 3E 20 00 BB 30 00 8A-04 02 05 88 07 46 47 43   .> ..0.......FGC
076A:0070  E2 F5 B4 4C CD 21 00 00-00 00 00 00 00 00 00 00   ...L.!..........
-g

Program terminated normally
-d 076a:0000
076A:0000  03 02 03 02 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0010  00 01 03 05 07 09 00 00-00 00 00 00 00 00 00 00   ................
076A:0020  02 04 06 08 0A 0C 00 00-00 00 00 00 00 00 00 00   ................
076A:0030  02 05 09 0D 11 15 00 00-00 00 00 00 00 00 00 00   ................
076A:0040  B8 6A 07 8E D8 A0 00 00-3A 06 02 00 75 24 8A 1E   .j......:...u$..
076A:0050  01 00 3A 1E 03 00 75 1A-F6 E3 8B C8 8D 36 10 00   ..:...u......6..
076A:0060  8D 3E 20 00 BB 30 00 8A-04 02 05 88 07 46 47 43   .> ..0.......FGC
076A:0070  E2 F5 B4 4C CD 21 00 00-00 00 00 00 00 00 00 00   ...L.!..........
```

```
Case ii: Incompatible matrices
row1 = 05H col1 = 02H
row2 = 05H col2 = 01H
```



```
-e 076a:0000
076A:0000  03.05   02.02   03.05   02.01

-d 076a:0000
076A:0000  05 02 05 01 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0010  00 01 03 05 07 09 00 00-00 00 00 00 00 00 00 00   ................
076A:0020  02 04 06 08 0A 0C 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  B8 6A 07 8E D8 A0 00 00-3A 06 02 00 75 24 8A 1E   .j......:...u$..
076A:0050  01 00 3A 1E 03 00 75 1A-F6 E3 8B C8 8D 36 10 00   ..:...u......6..
076A:0060  8D 3E 20 00 BB 30 00 8A-04 02 05 88 07 46 47 43   .> ..0.......FGC
076A:0070  E2 F5 B4 4C CD 21 00 00-00 00 00 00 00 00 00 00   ...L.!..........
-g

Program terminated normally
-d 076a:0000
076A:0000  05 02 05 01 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0010  00 01 03 05 07 09 00 00-00 00 00 00 00 00 00 00   ................
076A:0020  02 04 06 08 0A 0C 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  B8 6A 07 8E D8 A0 00 00-3A 06 02 00 75 24 8A 1E   .j......:...u$..
076A:0050  01 00 3A 1E 03 00 75 1A-F6 E3 8B C8 8D 36 10 00   ..:...u......6..
076A:0060  8D 3E 20 00 BB 30 00 8A-04 02 05 88 07 46 47 43   .> ..0.......FGC
076A:0070  E2 F5 B4 4C CD 21 00 00-00 00 00 00 00 00 00 00   ...L.!..........
```

## Result:

Program to add two matrices is assembled, executed and verified.

Aravind Kannan Rathinasabapathi
195001022

Date: 13-08-2021
Ex. No. 5

## 5B – Matrix Subtraction

### Aim:

To perform matrix subtraction.

### Algorithm:

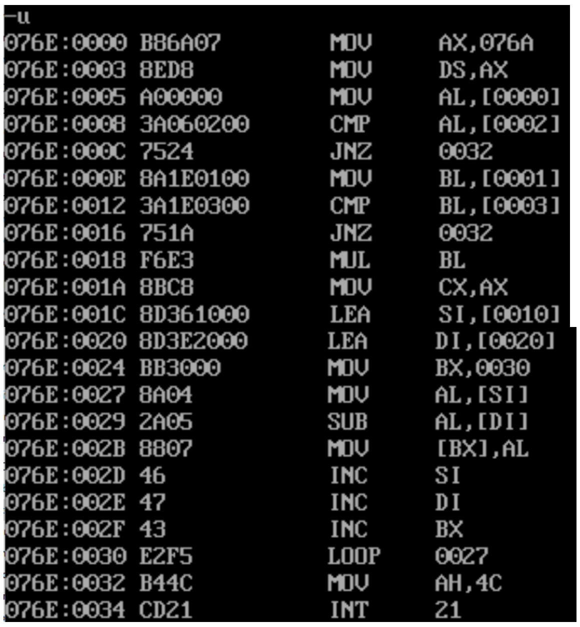
1. Initialise data and extra segment using their respective registers.
2. Load the base address of data segment(ds) and extra segment(es) using an intermediate accumulator register(ax) as direct memory transfer is not allowed in 8086.
3. Check for equality in number of rows and columns, else, terminate.
4. Calculate the length of the matrix, row * col, to run the loop, iter and store it in CX.
5. Load effective address of mat1, mat2 into SI, DI. Move the offset value of res into BX.
6. Move content at SI to AL, subtract it with content at DI. Move the result back to res using BX.
7. Increment SI, DI, BX.
8. Decrement CX and loop over the body, 6 and 7, until CX != 0.
9. Terminate the program.

Aravind Kannan Rathinasabapathi Date: 13-08-2021
195001022 Ex. No. 5

## Program:

| Program | Comment |
|---|---|
| `; 5b: Program to subtract 2 matrices` | Comment after ‘;’ |
| `assume cs: code, ds: data` | Map CS to code segment, DS to data segment |
| `data segment`<br>`    row1 db 03H`<br>`    col1 db 02H`<br>`    row2 db 03H`<br>`    col2 db 02H`<br>`    org 10H`<br>`    mat1 db 02H, 04H, 06H, 08H, 0aH, 0cH`<br>`    org 20H`<br>`    mat2 db 00H, 01H, 03H, 05H, 07H, 09H`<br>`    org 30H`<br>`    res db ?` | Initialise data segment and extra segment<br>db = define a byte<br>Initialise row1, col1, row2, col2<br>Initialise mat1, mat2<br>Define res |
| `data ends`<br>`code segment`<br>`start:  mov ax, data`<br>`        mov ds, ax` | Initialise code segment<br>Move the starting address of data segment in ax, then move ax to ds. |
| `        ; Check row1 == row2`<br>`        mov al, row1`<br>`        cmp al, row2`<br>`        jne term` | Load AL with row1, compare it against row2 to set flag registers. Jump to terminate, if zero flag is not set. |
| `        ; Check col1 == col2`<br>`        mov bl, col1`<br>`        cmp bl, col2`<br>`        jne term` | Load BL with col1, compare it against col2 to set flag registers. Jump to terminate, if zero flag is not set. |
| `        ; length of row major`<br>`        mul bl`<br>`        mov cx, ax` | Calculate the length of the matrix, row * col, to run the loop, iter and store it in CX. |
| `        ; ptr to operands and result`<br>`        lea si, mat1`<br>`        lea di, mat2`<br>`        mov bx, offset res` | Load effective address of mat1, mat2 into SI, DI. Move the offset value of res into BX. |
| `iter:   mov al, [si]`<br>`        sub al, [di]`<br>`        mov [bx], al`<br>`        inc si`<br>`        inc di`<br>`        inc bx`<br>`        loop iter` | Move content at SI to AL, subtract it with content at DI. Move the result back to res using BX. Increment SI, DI and BX. Decrement CX and loop until CX != 0 |
| `term:   mov ah, 4cH`<br>`        int 21H` | Set ah = 4cH<br>Call interrupt routine 21H for DOS, which terminates if ah = 4cH |
| `code ends`<br>`end start` | |

## Unassembled code:

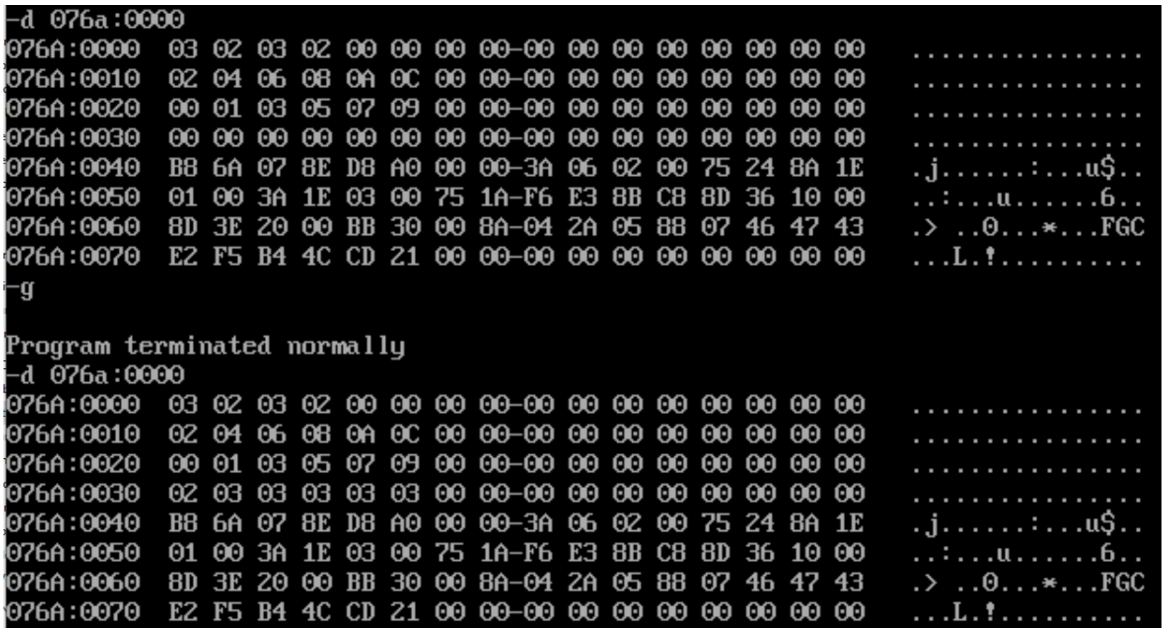


```
-u
076E:0000 B86A07        MOV     AX,076A
076E:0003 8ED8          MOV     DS,AX
076E:0005 A00000        MOV     AL,[0000]
076E:0008 3A060200      CMP     AL,[0002]
076E:000C 7524          JNZ     0032
076E:000E 8A1E0100      MOV     BL,[0001]
076E:0012 3A1E0300      CMP     BL,[0003]
076E:0016 751A          JNZ     0032
076E:0018 F6E3          MUL     BL
076E:001A 8BC8          MOV     CX,AX
076E:001C 8D361000      LEA     SI,[0010]
076E:0020 8D3E2000      LEA     DI,[0020]
076E:0024 BB3000        MOV     BX,0030
076E:0027 8A04          MOV     AL,[SI]
076E:0029 2A05          SUB     AL,[DI]
076E:002B 8807          MOV     [BX],AL
076E:002D 46            INC     SI
076E:002E 47            INC     DI
076E:002F 43            INC     BX
076E:0030 E2F5          LOOP    0027
076E:0032 B44C          MOV     AH,4C
076E:0034 CD21          INT     21
```

## Snapshot of sample input and output:

Case i: Compatible matrices

row1 = 03H col1 = 02H mat1 = [[02H, 04H], [06H, 08H], [0aH, 0cH]]
row2 = 03H col2 = 02H mat2 = [[00H, 01H], [03H, 05H], [07H, 09H]]
mat2 = [[02H, 03H], [03H, 03H], [03H, 03H]]

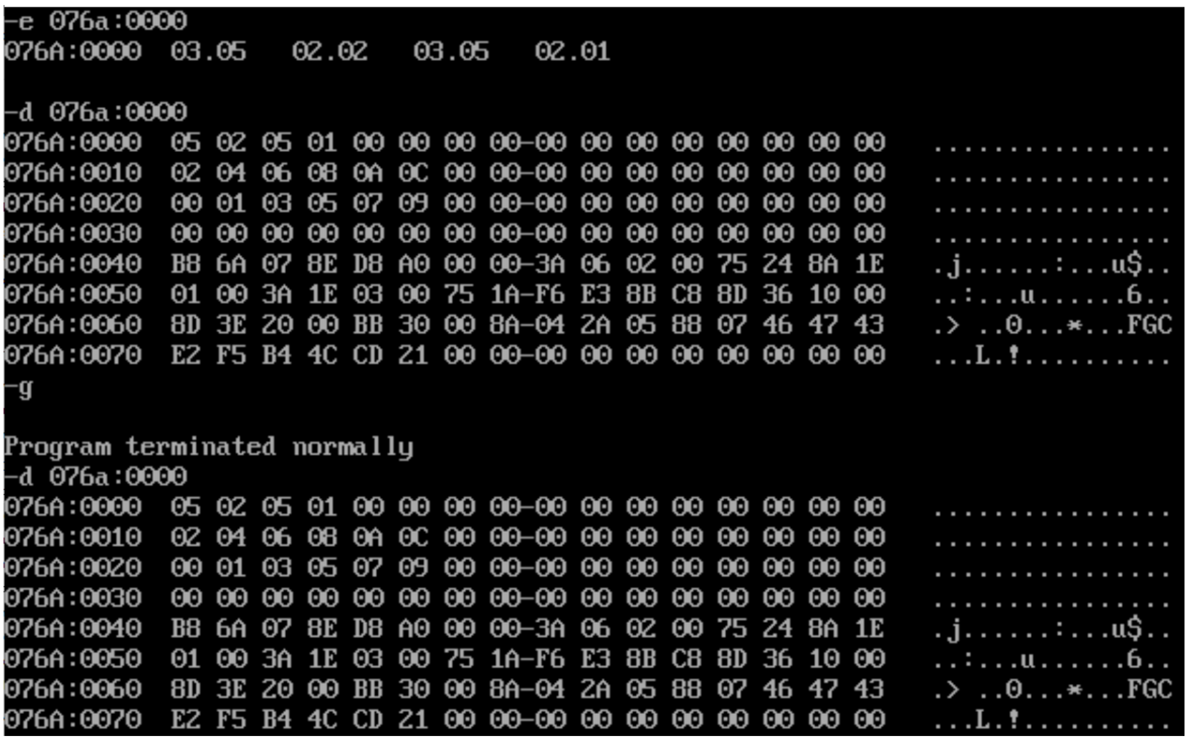


```
-d 076a:0000
076A:0000  03 02 03 02 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0010  02 04 06 08 0A 0C 00 00-00 00 00 00 00 00 00 00   ................
076A:0020  00 01 03 05 07 09 00 00-00 00 00 00 00 00 00 00   ................
076A:0030  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0040  B8 6A 07 8E D8 A0 00 00-3A 06 02 00 75 24 8A 1E   .j......:...u$..
076A:0050  01 00 3A 1E 03 00 75 1A-F6 E3 8B C8 8D 36 10 00   ..:...u......6..
076A:0060  8D 3E 20 00 BB 30 00 8A-04 2A 05 88 07 46 47 43   .> ..0...*...FGC
076A:0070  E2 F5 B4 4C CD 21 00 00-00 00 00 00 00 00 00 00   ...L.!..........
-g

Program terminated normally
-d 076a:0000
076A:0000  03 02 03 02 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0010  02 04 06 08 0A 0C 00 00-00 00 00 00 00 00 00 00   ................
076A:0020  00 01 03 05 07 09 00 00-00 00 00 00 00 00 00 00   ................
076A:0030  02 03 03 03 03 03 00 00-00 00 00 00 00 00 00 00   ................
076A:0040  B8 6A 07 8E D8 A0 00 00-3A 06 02 00 75 24 8A 1E   .j......:...u$..
076A:0050  01 00 3A 1E 03 00 75 1A-F6 E3 8B C8 8D 36 10 00   ..:...u......6..
076A:0060  8D 3E 20 00 BB 30 00 8A-04 2A 05 88 07 46 47 43   .> ..0...*...FGC
076A:0070  E2 F5 B4 4C CD 21 00 00-00 00 00 00 00 00 00 00   ...L.!..........
```

Aravind Kannan Rathinasabapathi
195001022 Date: 13-08-2021
Ex. No. 5

```
Case ii: Incompatible matrices
row1 = 05H col1 = 02H
row2 = 05H col2 = 01H
```



```
-e 076a:0000
076A:0000  03.05   02.02   03.05   02.01

-d 076a:0000
076A:0000  05 02 05 01 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0010  02 04 06 08 0A 0C 00 00-00 00 00 00 00 00 00 00   ................
076A:0020  00 01 03 05 07 09 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  B8 6A 07 8E D8 A0 00 00-3A 06 02 00 75 24 8A 1E   .j......:...u$..
076A:0050  01 00 3A 1E 03 00 75 1A-F6 E3 8B C8 8D 36 10 00   ..:...u......6..
076A:0060  8D 3E 20 00 BB 30 00 8A-04 2A 05 88 07 46 47 43   .> ..0...*...FGC
076A:0070  E2 F5 B4 4C CD 21 00 00-00 00 00 00 00 00 00 00   ...L.!..........
-g

Program terminated normally
-d 076a:0000
076A:0000  05 02 05 01 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0010  02 04 06 08 0A 0C 00 00-00 00 00 00 00 00 00 00   ................
076A:0020  00 01 03 05 07 09 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  B8 6A 07 8E D8 A0 00 00-3A 06 02 00 75 24 8A 1E   .j......:...u$..
076A:0050  01 00 3A 1E 03 00 75 1A-F6 E3 8B C8 8D 36 10 00   ..:...u......6..
076A:0060  8D 3E 20 00 BB 30 00 8A-04 2A 05 88 07 46 47 43   .> ..0...*...FGC
076A:0070  E2 F5 B4 4C CD 21 00 00-00 00 00 00 00 00 00 00   ...L.!..........
```

## Result:

Program to subtract two matrices is assembled, executed and verified.