Exp No: 5

Matrix operations

Date: 21/9/2020

Name: Gokhulnath T
Register Number: 185001051

Aim: To write assembly program to do following Matrix operations:

a) Matrix addition
b) Matrix subtraction

Procedure:

1. Install all the required file for executing MASM programs.(Masm, edit, link, debug etc..).
2. Write the assembly program in any editor before mounting the folder to the MASM.
3. Mount the folder that contains the assembly program with any name such as “d”.
    - mount d e:\masm
4. Create the object file of the assembly program using masm.
    - masm 16BITADD.asm
5. Use the link to create the executable file of the object file created from the above step.
    - Link 16BITADD.obj
6. Run the executable file using debug.
    - debug 16BITADD.exe
7. By un-assembling the program you can check the code segment of the program
    - u 076b:0100
8. To check the data memory segment, you can use the memory option to view the data stored.
    - d 076a:0000
9. To enter your own values, you can use the enter option which will prompt for new values.
    - e 076a:0000
10. To execute the program, you can use go option
    - G
11. After successful execution and termination of the program, you can check the result by checking the data memory segment
    - d 076a:0000
12. The result can be viewed in the respective address mentioned in the program.

5 a) Matrix addition

Algorithm:

a) Assign data to ax register
b) Load contents of memory location ax in register ds
c) Load contents of memory location row1 in register cl
d) Load contents of memory location row2 in register dl
e) Compare row1 and row2
f) If not equal exit else continue
g) Load contents of memory location col1 in register cl
h) Load contents of memory location col2 in register dl
i) Compare col1 and col2
j) If not equal exit else continue
k) Load contents of memory location row2 in register al
l) Multiply col1 and row2
m) Load contents of memory location ax in register cx
n) Load contents of memory location offset mat1 in register si
o) Load contents of memory location offset mat2 in register di
p) Load contents of memory location offset result in register dx
q) Here loop starts
r) Load contents of memory location [si]in register al
s) Add al and [di]
t) Load contents of memory location al in register [bx]
u) Increment si
v) Increment di
w) Increment bl
x) Loop again to here
y) Here loop ends
z) Load content 4ch termination code to ah register
aa) Stops execution of the program

Program:

Program	Comments
assume cs:code,ds:data,es:extra	Initializing the code, data and extra segments to assembler
data segment	Data segment
row1 db 02h	row1 is declared and initialized to 02h
row2 db 02h	row2 is declared and initialized to 02h
col1 db 04h	col1 is declared and initialized to 04h
col2 db 04h	col2 is declared and initialized to 04h
org 0010h	Mat1 data segment starting from address range 0010h
mat1 db 00h,12h,13h,14h,15h,16h,17h,18h	mat1 is declared and initialized to 00h,12h,13h,14h,15h,16h,17h,18h
org 0020h	Mat2 data segment starting from address range 0020h
mat2 db 11h,22h,33h,44h,55h,66h,77h,00h	Mat2 is declared and initialized to 11h,22h,33h,44h,55h,66h,77h,00h
org 0030h	result data segment starting from address range 0030h
result db 8 DUP(0)	result is declared and initialized to empty
data ends	
code segment	Code segment
org 0100h	assemble the code starting from address range 0100h
start: mov ax,data	Transferring the data from memory location data to ax
mov ds,ax	Transferring the data from memory location ax to ds
mov cl,row1	Transferring the data from memory location row1 to cl
mov dl,row2	Transferring the data from memory location row2 to dl
cmp cl,dl	Compare cl and dl
jne last	If not equal jump to last, else continue
mov cl,col1	Transferring the data from memory location col1 to cl

mov dl,col2	Transferring the data from memory location col2 to dl
cmp cl,dl	Compare cl and dl
jne last	If not equal jump to last, else continue
mov al,row2	Transferring the data from memory location row2 to al
mul cl	Multiply cl
mov cx,ax	Transferring the data from memory location ax to cx
mov si, offset mat1	Transferring the data from memory location offset mat1 to si
mov di, offset mat2	Transferring the data from memory location offset mat2 to di
mov bx, offset result	Transferring the data from memory location offset result to bx
here: mov al, [si]	Here loop starts Transferring the data from memory location [si] to al
add al, [di]	Add al => al+[di]
mov [bx], al	Transferring the data from memory location [bx] to al
inc si	Increment si
inc di	Increment di
inc bl	Increment bl
loop here	Loop continues
last: mov ah,4ch	Transferring the termination code 4ch to ah
int 21h	Termination
code ends	Code ends
end start	

Unassembled code:

```
D:\>debug 5A.EXE
-u
076E:0100 B86A07        MOV     AX,076A
076E:0103 8ED8          MOV     DS,AX
076E:0105 8A0E0000      MOV     CL,[0000]
076E:0109 8A160100      MOV     DL,[0001]
076E:010D 38D1          CMP     CL,DL
076E:010F 7528          JNZ     0139
076E:0111 8A0E0200      MOV     CL,[0002]
076E:0115 8A160300      MOV     DL,[0003]
076E:0119 38D1          CMP     CL,DL
076E:011B 751C          JNZ     0139
076E:011D A00100        MOV     AL,[0001]
-_
```

Sample Input and output:

Case1: if rows and columns are equal.

```
-d 076a:0000
076A:0000  02 02 04 04 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0010  00 12 13 14 15 16 17 18-00 00 00 00 00 00 00 00   ................
076A:0020  11 22 33 44 55 66 77 00-00 00 00 00 00 00 00 00   ."3DUfw.........
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  02 02 04 04 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0010  00 12 13 14 15 16 17 18-00 00 00 00 00 00 00 00   ................
076A:0020  11 22 33 44 55 66 77 00-00 00 00 00 00 00 00 00   ."3DUfw.........
076A:0030  11 34 46 58 6A 7C 8E 18-00 00 00 00 00 00 00 00   .4FXj|..........
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   ................
-_
```

Case2: if rows and columns are not equal.

```
-d 076a:0000
076A:0000  02 02 03 04 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0010  00 12 13 14 15 16 00 00-00 00 00 00 00 00 00 00   ................
076A:0020  11 22 33 44 55 66 77 00-00 00 00 00 00 00 00 00   ."3DUfw.........
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  02 02 03 04 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0010  00 12 13 14 15 16 00 00-00 00 00 00 00 00 00 00   ................
076A:0020  11 22 33 44 55 66 77 00-00 00 00 00 00 00 00 00   ."3DUfw.........
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   ................
-d_
```

Result:

Thus, the assembly program for matrix addition is written and executed.

5 b) Matrix subtraction

Algorithm:

a) Assign data to ax register
b) Load contents of memory location ax in register ds
c) Load contents of memory location row1 in register cl
d) Load contents of memory location row2 in register dl
e) Compare row1 and row2
f) If not equal exit else continue
g) Load contents of memory location col1 in register cl
h) Load contents of memory location col2 in register dl
i) Compare col1 and col2
j) If not equal exit else continue
k) Load contents of memory location row2 in register al
l) Multiply col1 and row2
m) Load contents of memory location ax in register cx
n) Load contents of memory location offset mat1 in register si
o) Load contents of memory location offset mat2 in register di
p) Load contents of memory location offset result in register dx
q) Here loop starts
r) Load contents of memory location [si]in register al
s) sub al and [di]
t) Load contents of memory location al in register [bx]
u) Increment si
v) Increment di
w) Increment bl
x) Loop again to here
y) Here loop ends
z) Load content 4ch termination code to ah register
aa) Stops execution of the program

Program:

Program	Comments
assume cs:code,ds:data,es:extra	Initializing the code, data and extra segments to assembler
data segment	Data segment
row1 db 02h	row1 is declared and initialized to 02h
row2 db 02h	row2 is declared and initialized to 02h
col1 db 04h	col1 is declared and initialized to 04h
col2 db 04h	col2 is declared and initialized to 04h
org 0010h	Mat1 data segment starting from address range 0010h
mat1 db 11h,22h,33h,44h,55h,66h,77h,88h	mat1 is declared and initialized to 11h,22h,33h,44h,55h,66h,77h,88h
org 0020h	Mat2 data segment starting from address range 0020h
mat2 db 00h,11h,22h,33h,44h,55h,66h,77h	Mat2 is declared and initialized to 00h,11h,22h,33h,44h,55h,66h,77h
org 0030h	result data segment starting from address range 0030h
result db 8 DUP(0)	result is declared and initialized to empty
data ends	
code segment	Code segment
org 0100h	assemble the code starting from address range 0100h
start: mov ax,data	Transferring the data from memory location data to ax
mov ds,ax	Transferring the data from memory location ax to ds
mov cl,row1	Transferring the data from memory location row1 to cl
mov dl,row2	Transferring the data from memory location row2 to dl
cmp cl,dl	Compare cl and dl
jne last	If not equal jump to last, else continue
mov cl,col1	Transferring the data from memory location col1 to cl

mov dl,col2	Transferring the data from memory location col2 to dl
cmp cl,dl	Compare cl and dl
jne last	If not equal jump to last, else continue
mov al,row2	Transferring the data from memory location row2 to al
mul cl	Multiply cl
mov cx,ax	Transferring the data from memory location ax to cx
mov si, offset mat1	Transferring the data from memory location offset mat1 to si
mov di, offset mat2	Transferring the data from memory location offset mat2 to di
mov bx, offset result	Transferring the data from memory location offset result to bx
here: mov al, [si]	Here loop starts Transferring the data from memory location [si] to al
sub al, [di]	sub al => al+[di]
mov [bx], al	Transferring the data from memory location [bx] to al
inc si	Increment si
inc di	Increment di
inc bl	Increment bl
loop here	Loop continues
last: mov ah,4ch	Transferring the termination code 4ch to ah
int 21h	Termination
code ends	Code ends
end start	

Unassembled code:

```
D:\>debug 5B.EXE
-u
076E:0100 B86A07        MOV     AX,076A
076E:0103 8ED8          MOV     DS,AX
076E:0105 8A0E0000      MOV     CL,[0000]
076E:0109 8A160100      MOV     DL,[0001]
076E:010D 38D1          CMP     CL,DL
076E:010F 7528          JNZ     0139
076E:0111 8A0E0200      MOV     CL,[0002]
076E:0115 8A160300      MOV     DL,[0003]
076E:0119 38D1          CMP     CL,DL
076E:011B 751C          JNZ     0139
076E:011D A00100        MOV     AL,[0001]
-
```

Sample Input and output:

Case1: if rows and columns are equal.

```
-d 076a:0000
076A:0000  02 02 04 04 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0010  11 22 33 44 55 66 77 88-00 00 00 00 00 00 00 00   ."3DUfw.........
076A:0020  00 11 22 33 44 55 66 77-00 00 00 00 00 00 00 00   .."3DUfw........
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  02 02 04 04 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0010  11 22 33 44 55 66 77 88-00 00 00 00 00 00 00 00   ."3DUfw.........
076A:0020  00 11 22 33 44 55 66 77-00 00 00 00 00 00 00 00   .."3DUfw........
076A:0030  11 11 11 11 11 11 11 11-00 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   ................
-_
```

Case2: if rows and columns are not equal.

```
-d 076a:0000
076A:0000  02 02 03 04 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0010  11 22 33 44 55 66 00 00-00 00 00 00 00 00 00 00   ."3DUf..........
076A:0020  00 11 22 33 44 55 66 77-00 00 00 00 00 00 00 00   .."3DUfw........
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  02 02 03 04 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0010  11 22 33 44 55 66 00 00-00 00 00 00 00 00 00 00   ."3DUf..........
076A:0020  00 11 22 33 44 55 66 77-00 00 00 00 00 00 00 00   .."3DUfw........
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:

Thus, the assembly program for matrix subtraction is written and executed.