

# MATRIX OPERATIONS

Exp No: 5

Name: Srinithyee S K

Date: 09-10-20

Register Number: 185001166

---

## **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:
  - i. Add values at SI and DI and push it into the stack.
  - ii. Increment SI and DI.
  - iii. Decrement CL.
15. Move SI to end of resultant matrix.
16. Till CL goes to zero:
  - i. Pop the value from top of the stack and put it at SI.
  - ii. 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	Exiting if not same.
al, bl jne	
break	
mov al, col1	Comparing column count of both matrices.
mov bl, col2	
cmp al, bl	Exiting if not same.
jne break	
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	AL has the value of row1 * col1.
mul bl	
mov len, al	Finding no. of elements in the matrix.
mov cl, len	Clear CH.
mov ch, 00h	Clear AX.
mov ax, 0000h	
loop: mov al, [si]	Pushing each element-wise sum into stack
mov ah, 00h	
mov bl, [di]	Add the 2 elements from each matrix.
mov bh, 00h	
add ax, bx	

push ax inc si inc di	Move to next element in matrix 1. Move to next element in matrix 2.
dec cx jz prewrk jmp loopr	Decrement counter by 1. If addition is over, jump to “prewrk” Repeat addition for all elements.
prewrk:        mov si, offset resi + 0001h mov cl, len mov ch, 00h add si, cx  retloop:        pop ax  mov [si], al dec si mov [si], ah dec si dec cx  jz break jmp retloop	Set the SI to store values in result matrix “resi” properly. Set counter to length of the matrix. Clear CH. Set SI to point to the last location of the matrix.  Popping each element from stack into resultant matrix.  Decrement SI.  Decrement counter by 1.
break: mov ah, 4ch int 21h code ends end start	Stop popping if all elements are popped (CX = 0) Pop the next element and put it in the matrix.   Interrupt the process with return code and exit.

## 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  #$.!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  #$.!DW".....
076A:0010 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
076A:0020 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
076A:0030 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
076A:0040 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
076A:0050 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
076A:0060 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
076A:0070 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
-g
Program terminated normally
-d 076A:0000
076A:0000 23 24 55 11 21 44 57 22-02 02 02 02 04 00 00 00  #$.!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:
  - i. Subtract values at SI and DI and push it into the stack.
  - ii. Increment SI and DI.
  - iii. Decrement CL.
15. Move SI to end of resultant matrix.
16. Till CL goes to zero:
  - i. Pop the value from top of the stack and put it at SI.
  - ii. 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	Exiting if not same.
al, bl jne	
break	
mov al, col1	Comparing column count of both matrices.
mov bl, col2	
cmp al, bl	Exiting if not same.
jne break	
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	AL has the value of row1 * col1.
mul bl	
mov len, al	Finding no. of elements in the matrix.
mov cl, len	Clear CH.
mov ch, 00h	Clear AX.
mov ax, 0000h	
loop: mov al, [si]	Pushing each element-wise sum into stack
mov ah, 00h	
mov bl, [di]	Subtract the 2 elements from each matrix.
mov bh, 00h sub	
ax, bx	

push ax inc si inc di	Move to next element in matrix 2. Move to next element in matrix 1.
dec cx jz prewrk jmp loopwr	Decrement counter by 1. If addition is over, jump to “prewrk” Repeat addition for all elements.
prewrk:        mov si, offset resi + 0001h mov cl, len mov ch, 00h add si, cx add si, cx  retloop:        pop ax  mov [si], al dec si mov [si], ah dec si   dec cx  jz break jmp retloop  break: mov ah, 4ch int 21h code ends end start	Set the SI to store values in result matrix “resi” properly. Set counter to length of the matrix. Clear CH. Set SI to point to the last location of the matrix.  Popping each element from stack into resultant matrix.  Decrement SI.  Decrement counter by 1.  Stop popping if all elements are popped (CX = 0) Pop the next element and put it in the matrix.  Interrupt the process with return code and exit.



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

# **SORTING**

Exp No: 6

Name: Srinithyee S K

Date: 11-10-20

Register Number: 185001166

---

## **AIM:**

To write assembly language programs to perform the following experiments:

1. Ascending order sorting using Bubble Sort.
2. Descending order sorting using Bubble Sort.

## PROGRAM 1: SORTING IN ASCENDING ORDER

### ALGORITHM:

1. Begin.
2. Declare the data segment.
3. Initialize data segment with array and its length len.
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. Move the length len to CH register.
10. Till CH goes to zero:
  - a. Load SI with offset of list.
  - b. Move the length len to CL register.
  - c. Till CL goes to zero:
    - i. Compare values at SI and SI+1 address.
    - ii. If value at SI > value at SI+1, exchange them.
    - iii. Increment SI.
    - iv. Decrement CL.
  - d. Decrement CH.
11. Introduce an interrupt for safe exit. (INT 21h)
12. Close the code segment.
13. End.

PROGRAM	COMMENTS
assume cs:code, ds:data	Declare code and data segment.
data segment	Initialize data segment with values.
list db 05h, 04h, 03h, 02h, 01h	Stores the list of elements.
len db 04h	Stores the length of the above array.
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 ch, len	
outer: mov si, offset list	Pointer at first element.
mov cl, len	Inner loop count.
inner: mov al, [si]	
mov ah, [si+1]	
cmp ah, al	Compare by AL – AH.
jnc skip	Skip if no carry occurred on AL – AH.
xchg al, ah	Exchange register contents.
mov [si], ax	Copy back moved contents to data segment (AL -> [SI], AH -> [SI + 1])
skip: inc si	Go to next element.
dec cl	Decrement inner loop count.
jnz inner	Restart inner loop.
dec ch	Decrement outer loop count.
jnz outer	Restart outer loop.
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:\>LINK ASCSORT.OBJ;

Microsoft Object Linker V2.01 (Large)
(C) Copyright 1982, 1983 by Microsoft Inc.

Warning: No STACK segment

There was 1 error detected.

Q:\>DEBUG ASCSORT.EXE
-u
076B:0100 B86A07      MOV     AX,076A
076B:0103 8ED8        MOV     DS,AX
076B:0105 8A2E0500     MOV     CH,[0005]
076B:0109 BE0000      MOV     SI,0000
076B:010C 8A0E0500     MOV     CL,[0005]
076B:0110 8A04        MOV     AL,[SI]
076B:0112 8A6401      MOV     AH,[SI+01]
076B:0115 38C4        CMP     AH,AL
076B:0117 7304        JNB     011D
076B:0119 86C4        XCHG    AL,AH
076B:011B 8904        MOV     [SI],AX
076B:011D 46          INC     SI
076B:011E FEC9        DEC     CL
-
```

## SAMPLE I/O SNAPSHOT:

```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Progra...
076B:011B 8904      MOV     [SI],AX
076B:011D 46          INC     SI
076B:011E FEC9      DEC     CL
-d 076A:0000
076A:0000 05 04 03 02 01 04 00 00-00 00 00 00 00 00 00 00 .....
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 01 02 03 04 05 04 00 00-00 00 00 00 00 00 00 00 .....
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 .....
-
```

## PROGRAM 2: SORTING IN DESCENDING ORDER

### ALGORITHM:

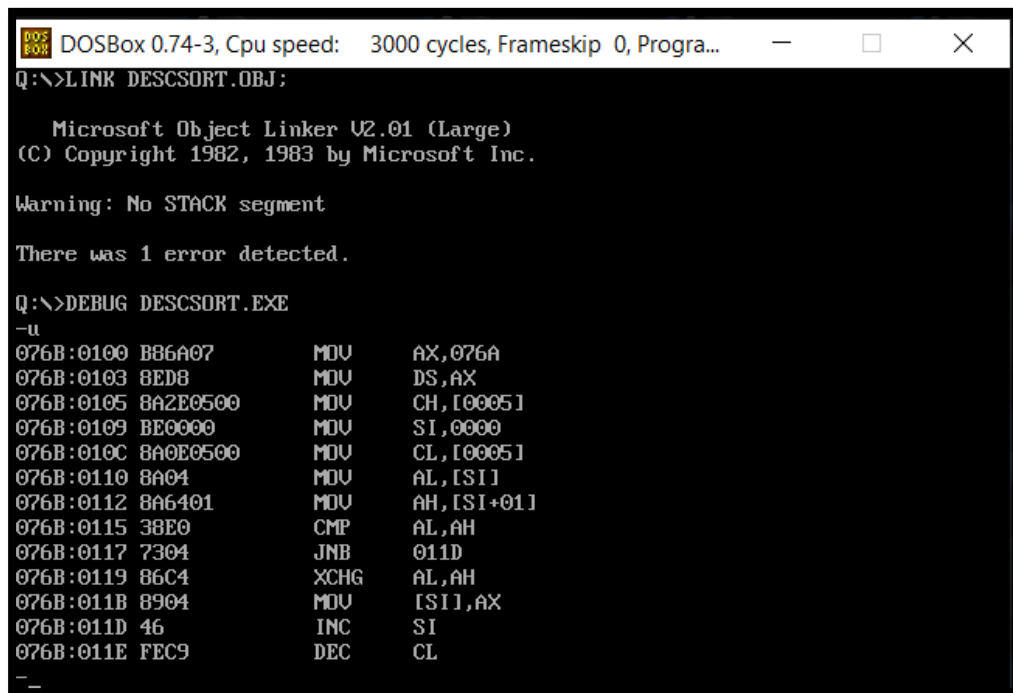
- 1.
2. Begin.
3. Declare the data segment.
4. Initialize data segment with array and its length len.
5. Close the data segment.
6. Declare the code segment.
7. Set a preferred offset (preferably 100)
8. Load the data segment content into AX register.
9. Transfer the contents of AX register to DS register.
10. Move the length len to CH register.
11. Till CH goes to zero:
  - a. Load SI with offset of list.
  - b. Move the length len to CL register.
  - c. Till CL goes to zero:
    - i. Compare values at SI and SI+1 address.
    - ii. If value at SI < value at SI+1, exchange them.
    - iii. Increment SI.
    - iv. Decrement CL.
  - d. Decrement CH.
12. Introduce an interrupt for safe exit. (INT 21h)
13. Close the code segment.
14. End.



PROGRAM	COMMENTS
assume cs:code, ds:data	Declare code and data segment.
data segment	Initialize data segment with values.
list db 05h, 04h, 03h, 02h, 01h	Stores the list of elements.
len db 04h	Stores the length of the above array.
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 ch, len	
outer: mov si, offset list	Pointer at first element.
mov cl, len	Inner loop count.
inner: mov al, [si]	
mov ah, [si+1]	
cmp al, ah	Compare by AH – AL.
jnc skip	Skip if no carry occurred on AH – AL.
xchg al, ah	Exchange register contents.
mov [si], ax	Copy back moved contents to data segment (AL -> [SI], AH -> [SI + 1])
skip: inc si	Go to next element.
dec cl	Decrement inner loop count.
jnz inner	Restart inner loop.
dec ch	Decrement outer loop count.
jnz outer	Restart outer loop.
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:\>LINK DESC SORT.OBJ;

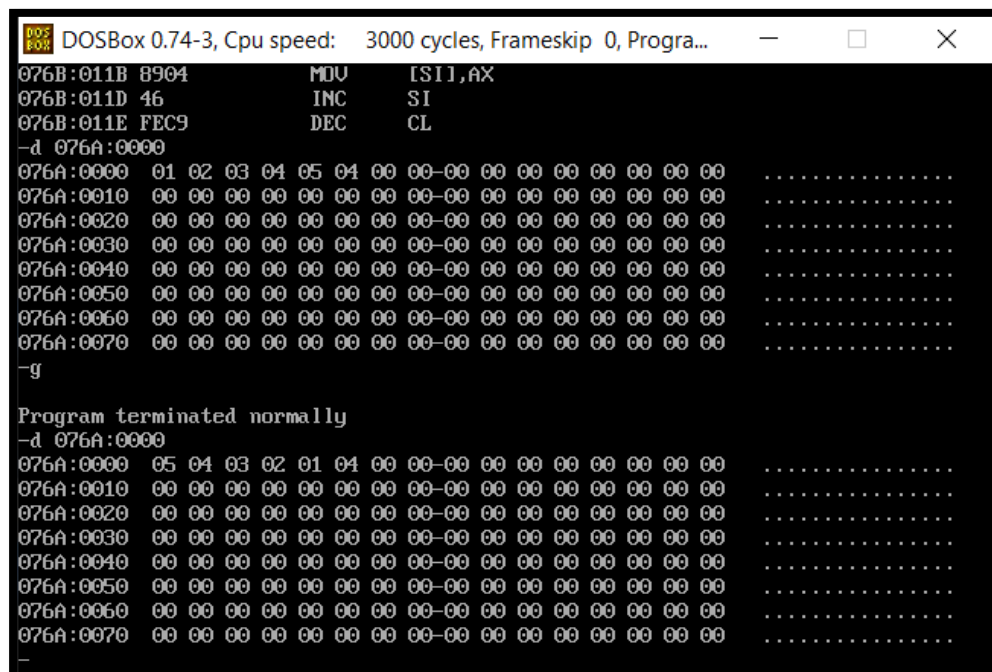
Microsoft Object Linker V2.01 (Large)
(C) Copyright 1982, 1983 by Microsoft Inc.

Warning: No STACK segment

There was 1 error detected.

Q:\>DEBUG DESC SORT.EXE
-u
076B:0100 B86A07      MOV     AX,076A
076B:0103 BED8        MOV     DS,AX
076B:0105 8A2E0500     MOV     CH,[0005]
076B:0109 BE0000      MOV     SI,0000
076B:010C 8A0E0500     MOV     CL,[0005]
076B:0110 8A04        MOV     AL,[SI]
076B:0112 8A6401      MOV     AH,[SI+01]
076B:0115 3BE0        CMP     AL,AH
076B:0117 7304        JNB     011D
076B:0119 86C4        XCHG    AL,AH
076B:011B 8904        MOV     [SI],AX
076B:011D 46          INC     SI
076B:011E FEC9        DEC     CL
-
```

### SAMPLE I/O SNAPSHOT:



```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Progra...
076B:011B 8904      MOV     [SI],AX
076B:011D 46          INC     SI
076B:011E FEC9      DEC     CL
-d 076A:0000
076A:0000 01 02 03 04 05 04 00 00-00 00 00 00 00 00 00 00 .....
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 05 04 03 02 01 04 00 00-00 00 00 00 00 00 00 00 .....
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 .....
```

### RESULT:

The assembly level programs were written to perform the above specified sorting functions and the output was verified.

# BCD ADDITION AND SUBTRACTION

Exp No: 7

Name: Srinithyee S K

Date: 12-10-20

Register Number: 185001166

---

## **AIM:**

To write assembly language programs to perform the following BCD arithmetic operations:

3. BCD Addition.
4. BCD Subtraction.

## PROGRAM 1: BCD ADDITION

### ALGORITHM:

1. Begin.
2. Declare the data segment.
3. Initialize data segment with the 2 BCD numbers and variables for storing their sum and carry.
4. Close the data segment.
5. Declare the code segment.
6. Set a preferred offset (preferably 100h)
7. Load the data segment content into AX register.
8. Transfer the contents of AX register to DS register.
9. Move the contents of the two numbers num1 and num2 to AL and BL register.
10. Add them and store the value in AL.
11. Move the contents of AL to sum.
12. Perform decimal adjust after addition on AL to get BCD result (HEX to BCD)
13. Check if the above adjustment produced a carry.
  - a. If carry was produced, set the variable carry to 1.
  - b. Else, continue.
14. Transfer the adjusted addition result to the variable sum.
15. Introduce an interrupt for safe exit. (INT 21h)
16. Close the code segment.
17. End.

PROGRAM	COMMENTS
assume cs:code, ds:data	Declare code and data segment.
data segment	Initialize data segment with values.
num1 db 26h	Stores the first BCD number.
num2 db 99h	Stores the second BCD number.
res db ?	Variable to store the sum of the 2 numbers.
carry db ?	Variable to store the carry of the above sum.
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, num1	Copy num1 to AL.
mov bl, num2	Copy num2 to BL.
mov cl, 00h	Clear CL register.
add al, bl	AL = AL + BL
daa	Adjust HEX result to BCD after subtraction.
jnc resume	If carry was not produced, jump to “resume”.
inc cl	Increment CL register by 1.
resume: mov res, al	Transfer AL contents to variable res.
mov carry, cl	Transfer CL contents to variable carry.
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...
Microsoft Object Linker V2.01 (Large)
(C) Copyright 1982, 1983 by Microsoft Inc.

Warning: No STACK segment

There was 1 error detected.

Q:\>DEBUG BCDADD.EXE;
-u
076B:0100 B86A07      MOV     AX,076A
076B:0103 8ED8        MOV     DS,AX
076B:0105 A00000      MOV     AL,[0000]
076B:0108 8A1E0100    MOV     BL,[0001]
076B:010C 02C3        ADD     AL,BL
076B:010E 27          DAA
076B:010F A20200      MOV     [0002],AL
076B:0112 B000        MOV     AL,00
076B:0114 12C0        ADC     AL,AL
076B:0116 A20300      MOV     [0003],AL
076B:0119 B44C        MOV     AH,4C
076B:011B CD21        INT     21
076B:011D 7701        JA      0120
076B:011F 40          INC     AX
_
```

### SAMPLE I/O SNAPSHOT:

```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Progra...
076B:011B CD21      INT     21
076B:011D 7701      JA      0120
076B:011F 40        INC     AX
-d 076A:0000
076A:0000 26 99 00 00 00 00 00 00-00 00 00 00 00 00 00 00  &.....
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 26 99 25 01 00 00 00 00-00 00 00 00 00 00 00 00  &.%.....
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  ..
_
```

## PROGRAM 1: BCD SUBTRACTION

### ALGORITHM:

1. Begin.
2. Declare the data segment.
3. Initialize data segment with the 2 BCD numbers and variables for storing their difference (diff) and sign.
4. Close the data segment.
5. Declare the code segment.
6. Set a preferred offset (preferably 100h)
7. Load the data segment content into AX register.
8. Transfer the contents of AX register to DS register.
9. Move the contents of the two numbers num1 and num2 to AL and BL register.
10. Subtract them and store the value in AL.
11. Transfer the contents of AL to diff.
12. If carry flag is set: (Performing 10's complement)
  - a. Set sign as 01h.
  - b. Move the contents of diff to BL register.
  - c. Move 99h to AL register.
  - d. Subtract BL from AL and store the value in AL register.
  - e. Move 01h to BL register.
  - f. Add AL and BL.
  - g. Perform decimal adjust on the addition in AL. (HEX to BCD).
  - h. Transfer the contents of AL to diff.
13. Introduce an interrupt for safe exit. (INT 21h)
14. Close the code segment.
15. End.



PROGRAM	COMMENTS
assume cs:code, ds:data	Declare code and data segment.
data segment	Initialize data segment with values.
num1    db    26h	Stores the first BCD number.
num2    db    99h	Stores the second BCD number.
diff    db    ?	Variable to store the difference of the 2 numbers.
sign    db    ?	Variable to store the sign of the above difference.
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, num1	Copy num1 to AL.
mov  bl, num2	Copy num2 to BL.
sub  al, bl	AL = AL – BL
das	Adjust HEX result to BCD after subtraction.
mov  diff, al	Transfer AL contents to diff.
jnc  break	If carry was not produced, jump to “break”.
mov  sign, 01h	If carry was produced, set sign to 1.
mov  al, 99h	Set AL = 99h to perform 9’s complement.
mov  bl, diff	Transfer diff to BL.
sub  al, bl	AL = 99h – BL (9’s complement)
mov  bl, 01h	Set BL = 01h.
add  al, bl	AL = AL + BL
daa	AL value is decimal adjusted after addition (HEX to BCD)
mov  diff, al	Transfer AL contents to diff.
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:\>LINK BCDSUB.OBJ;

Microsoft Object Linker V2.01 (Large)
(C) Copyright 1982, 1983 by Microsoft Inc.

Warning: No STACK segment

There was 1 error detected.

Q:\>DEBUG BCDSUB.EXE
-u
076B:0100 B86A07      MOV     AX,076A
076B:0103 8ED8        MOV     DS,AX
076B:0105 A00000        MOV     AL,[0000]
076B:0108 8A1E0100     MOV     BL,[0001]
076B:010C 2AC3        SUB     AL,BL
076B:010E 2F          DAS
076B:010F A20200        MOV     [0002],AL
076B:0112 7315        JNB     0129
076B:0114 C606030001   MOV     BYTE PTR [0003],01
076B:0119 B099        MOV     AL,99
076B:011B 8A1E0200     MOV     BL,[0002]
076B:011F 2AC3        SUB     AL,BL
-
```

## SAMPLE I/O SNAPSHOT:

```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Progra...
076B:0119 B099        MOV     AL,99
076B:011B 8A1E0200     MOV     BL,[0002]
076B:011F 2AC3        SUB     AL,BL
-d 076A:0000
076A:0000 15 35 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .5.....
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 15 35 20 01 00 00 00 00-00 00 00 00 00 00 00 00 .5 .....
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 .....
-
```

**RESULT:**

The assembly level programs were written to perform the above specified BCD arithmetic operations and their output was verified.