Exp No: 5 **Name : Sowmya R**
Date: 09.10.2020 **Reg No. 185001159**

Matrix operations

---

1. Aim:

To execute assembly language programs for performing matrix addition and subtraction with 8 bit numbers using MASM software.

2. Procedure for executing MASM

- Run the Dosbox software and mount the masm folder to a drive (say D) inside
- Dosbox.
- Shift to the mounted drive by specifying the drive name (eg D:)
- Use the edit command to modify the 8086 program and save it with the extension
- ".asm" inside the masm folder.
- Execute the command " masm filename.asm" to assemble the file and create an
- object file.This creates a "filename.obj" file in your masm folder.
- Use the link command – "link filename.obj" to link the libraries and create a
- "filename.exe" file.
- Finally, use the debug command – "debug filename.exe" to execute the program.
- The following options can be used inside the debug console to perform different Operations:
    - ? – displays the help menu and lists all the options that can be used.
    - u(unassemble) – unassembles machine instructions into 8086 Assembly code.
    - d(dump) - Displays the contents of a block of memory.
    - e(enter) - Used to enter data or instructions (as machine code) directly into
    - Memory locations
    - g(go) - Go is used to run a program and set breakpoints in the program's
    - code. If you use 'g' all by itself, execution will begin at whatever location is
    - pointed to by the CS:IP registers.
    - q(quit) – Immediately quits the program

- Use command "exit" to close the dosbox command prompt.

Matrix Addition

Algorithm:

1. Start
2. Move the address of data segment to register DS .
3. Compare the dimensions of the two matrices (row1 = rows 2 and col1= col2).
4. If the dimensions doesn't match exit the program using jump instructions.
5. Calculate the number of elements in the resultant matrix by multiplying row and col value of either of the matrices and transfer it to CX reg to serve as a counter.
6. Move the offset of mat1 and mat2 to SI and DI respectively.
7. Move the offset of the result to BX register
8. Add the contents of [SI] and [DI] and store the result in [BX] using a loop until CX value becomes 0 and increment SI ,DI and BX for each iteration.
9. Terminate the program

Program:

Program	Comments
assume cs:code,ds:data	Data and code segments are initialized
data segment	
row1 db 03h	row1← 03h
row2 db 03h	row2 ← 03h
col1 db 03h	col1 ← 03h
col2 db 03h	col2 ← 03h
org 0010h	
mat1 db 07h,04h,06h,03h,01h,01h,01h,03h,04h	mat1 is declared and initialized and address is set to 0010h
org 0020h	
mat2 db 03h,02h,05h,07h,04h,03h,01h,08h,06h	mat2 is declared and initialized and address is set to 0020h
org 0030h	
result db ?	result is declared and address is set to 0030h
data ends	
code segment	Code segment begins
org 0100h	Originating address is set to 0100h
start: mov ax,data	AX ← data
mov ds,ax	DS ← AX
mov cl,row1	CL ← row1
mov dl,row2	DL ← row2

cmp cl,dl	Compare CL, DL
jne over	if not equal jump to label over
mov cl,col1	CL ← col1
mov dl,col2	DL ← col2
cmp cl,dl	Compare CL and DL
jne over	if not equal jump to label over
mov al,row2	AL ← row2
mul cl	AX ← CL x AL
mov cx,ax	CX ← AX
mov si, offset mat1	Offset of the mat1,mat2,result are
mov di, offset mat2	transferred to SI, DI and BX respectively
mov bx, offset result	
here: mov ah,00h	AH ← 00h
mov al, [si]	
add al, [di]	Add [SI] and [DI] , if there is carry
jnc here1	increment AH register else jump to here1
inc ah	
here1: mov [bx], al	[BX] ← AL
inc si	SI ← SI + 1
inc di	DI ← DI +1
inc bx	BX ← BX + 1
loop here	Repeat till CX becomes 0
over: mov ah,4ch	
int 21h	Program terminates
code ends	
end start	

Unassembled code:

```
-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 752D          JNZ     013E
076E:0111 8A0E0200      MOV     CL,[0002]
076E:0115 8A160300      MOV     DL,[0003]
076E:0119 38D1          CMP     CL,DL
076E:011B 7521          JNZ     013E
076E:011D A00100        MOV     AL,[0001]
-
```

Sample Input and Output:

```
D:\>debug 5a.exe
-d 076a:0000
076A:0000  03 03 02 02 00 00 00 00-00 00 00 00 00 00 00 00    ................
076A:0010  03 01 01 01 03 04 00 00-00 00 00 00 00 00 00 00    ................
076A:0020  07 04 03 01 08 06 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  03 03 02 02 00 00 00 00-00 00 00 00 00 00 00 00    ................
076A:0010  03 01 01 01 03 04 00 00-00 00 00 00 00 00 00 00    ................
076A:0020  07 04 03 01 08 06 00 00-00 00 00 00 00 00 00 00    ................
076A:0030  0A 05 04 02 0B 0A 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    ................
```

Matrix Subtraction

Algorithm:

1. Start
2. Move the address of data segment to register DS .
3. Compare the dimensions of the two matrices (row1 = rows 2 and col1= col2).
4. If the dimensions doesn't match exit the program using jump instructions.
5. Calculate the number of elements in the resultant matrix by multiplying row and col value of either of the matrices and transfer it to CX reg to serve as a counter.
6. Move the offset of the mat1 and mat2 to SI and DI respectively.
7. Move the offset of the result to BX register
8. Repeatedly subtract the contents of [SI] and [DI] and store the result in [BX] by looping until CX = 0 and increment SI ,DI and BX in every iteration.
9. Terminate the program.

Program:

Program	Comments
assume cs:code,ds:data	Data and code segment is initialized
data segment	
row1 db 03h	row1← 03h
row2 db 03h	Row2 ← 03h
col1 db 03h	Col1 ← 03h
col2 db 03h	Col2 ← 03h
org 0010h	
mat1 db 07h,04h,06h,03h,06h,05h,01h,03h,04h	mat1 is declared and initialized and address is set to 0010h
org 0020h	
mat2 db 03h,02h,05h,07h,04h,03h,01h,08h,06h	mat2 is declared and initialized and address is set to 0020h
org 0030h	
result db ?	result is declared and address is set to 0030h
data ends	
code segment	Code segment begins
org 0100h	Originating address is set to 0100h
start: mov ax,data	AX ← data
mov ds,ax	DS ← AX

mov cl,row1	CL ← row1
mov dl,row2	DL ← row2
cmp cl,dl	Compare CL and DL
jne over	if not equal jump to label over
mov cl,col1	CL ← col1
mov dl,col2	DL ← col2
cmp cl,dl	Compare CL and DL
jne over	if not equal jump to label over
mov al,row2	AL ← row2
mul cl	CL ← CL x AL
mov cx,ax	CX ← AX
mov si, offset mat1	Offset of the mat1,mat2,result are
mov di, offset mat2	transferred to SI, DI and BX respectively
mov bx, offset result	
here: mov ah,00h	AH ← 00h
mov al, [si]	Subtract [SI] and [DI] , if there is carry
sub al, [di]	increment AH register else jump to here1
jnc here1	
inc ah	
here1: mov [bx], al	[BX] ← AL
inc si	SI ← SI + 1
inc di	DI ← DI +1
inc bx	BX ← BX + 1
loop here	Repeat till CX becomes 0
over: mov ah,4ch	
int 21h	Program terminates
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 752D          JNZ     013E
076E:0111 8A0E0200      MOV     CL,[0002]
076E:0115 8A160300      MOV     DL,[0003]
076E:0119 38D1          CMP     CL,DL
076E:011B 7521          JNZ     013E
076E:011D A00100        MOV     AL,[0001]
-
```

Sample Input and Output:

```
-d 076a:0000
076A:0000  03 03 02 02 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0010  03 06 05 01 03 04 00 00-00 00 00 00 00 00 00 00   ................
076A:0020  07 04 03 01 08 06 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  03 03 02 02 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0010  03 06 05 01 03 04 00 00-00 00 00 00 00 00 00 00   ................
076A:0020  07 04 03 01 08 06 00 00-00 00 00 00 00 00 00 00   ................
076A:0030  FC 02 02 00 FB FE 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, assembly language programs for matrix addition and subtraction have been executed and the output has been verified.

Exp no: 6

Sorting

Aim:

To execute assembly language programs for performing sorting in ascending and descending order using MASM software.

Sorting in ascending order

Algorithm:

1. Start
2. Move the address of data segment to register DS .
3. Move the length of the array to CH register(outer loop).
4. Make a label loop1 for outer loop.
5. Move the offset of the array to the SI register.
6. Move the length of the array to CL register(inner loop).
7. Make a label loop2 for inner loop.
8. Move content of SI to AL and SI + 1 to AH.
9. Compare AH and AL , if there is no carry produced jump to label skip , else swap the two values using XCHG instruction.
10. Move the content of AL to [SI] and content of AH to [SI+1].
11. Make a label skip to skip the swapping process if the elements are in correct order.
12. Increment SI and decrement CL ,if CL is not zero jump to label loop2.
13. Decrement CH , if CH is not zero jump to label loop1.
14. Terminate the program.

Program:

Program	Comments
assume cs:code,ds:data	Data and code segments are initialized

data segment array db 05h,03h,02h,07h,06h,01h,00h,09h,08h,04h data ends	
code segment org 0100h	Code segment begins Originating address is set to 0100h

start : mov ax,data
  mov ds,ax
  mov ch , 09h
	AX ←data
	DS ← AX
	CH ← 09h (outer loop count)

loop1 : mov si ,offset array mov cl , 09h	Move the offset of array to SI CL ← 09h

loop2 : mov al , [si]
  mov ah , [si+1]
  cmp ah , al
  jnc skip
  xchg al,ah
  mov [si] ,al
  mov [si+1],ah
	Move [SI] to AL and [SI+1] to AH registers
	
	Compare AH and AL
	If there is no carry(i.e they are in correct order) jump to skip else swap AL and AH
	Move AL to [SI] and AH to [SI+1] again

skip: inc si
  dec cl
  jnz loop2
  dec ch
  jnz loop1
	SI ← SI + 1
	CL ← CL - 1
	if CL is not zero jump to loop2.
	CH ← CH - 1
	if CH is not zero jump to loop1

mov ah,4ch int 21h code ends end start	Program terminates

Unassembled code:

```
D:\>debug 6a.exe
-u
076B:0100 B86A07        MOV     AX,076A
076B:0103 8ED8          MOV     DS,AX
076B:0105 B509          MOV     CH,09
076B:0107 BE0000        MOV     SI,0000
076B:010A B109          MOV     CL,09
076B:010C 8A04          MOV     AL,[SI]
076B:010E 8A6401        MOV     AH,[SI+01]
076B:0111 38C4          CMP     AH,AL
076B:0113 7307          JNB     011C
076B:0115 86C4          XCHG    AL,AH
076B:0117 8804          MOV     [SI],AL
076B:0119 886401        MOV     [SI+01],AH
076B:011C 46            INC     SI
076B:011D FEC9          DEC     CL
076B:011F 75EB          JNZ     010C
-
```

Sample Input and Output:

```
-d 076a:0000
076A:0000  05 03 02 07 06 01 00 09-08 04 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  00 01 02 03 04 05 06 07-08 09 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   ................
-
```

Sorting in descending order

Algorithm:

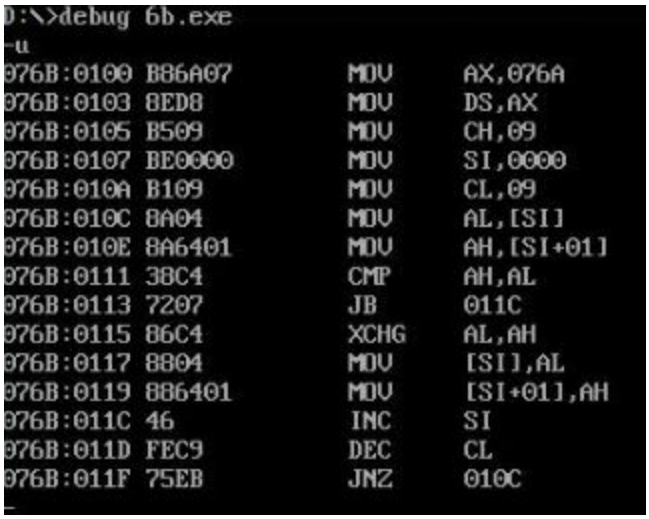
1. Start
2. Move the address of the data segment to register DS .
3. Move the length of the array to CH register(outer loop).
4. Make a label loop1 for outer loop.
5. Move the offset of the array to the SI register.
6. Move the length of the array to the CL register(inner loop).
7. Make a label loop2 for the inner loop.
8. Move content of SI to AL and SI + 1 to AH.
9. Compare AH and AL , if there is a carry produced jump to label skip else
10. Swap the two values using XCHG instruction.
11. Move the content of AL to [SI] and content of AH to [SI+1].
12. Make a label skip to skip the swapping process if the elements are in correct order.
13. Increment SI and decrement CL , if CL is not zero jump to label loop2.
14. Decrement CH ,if CH is not zero jump to label loop1.
15. Terminate the program.

Program:

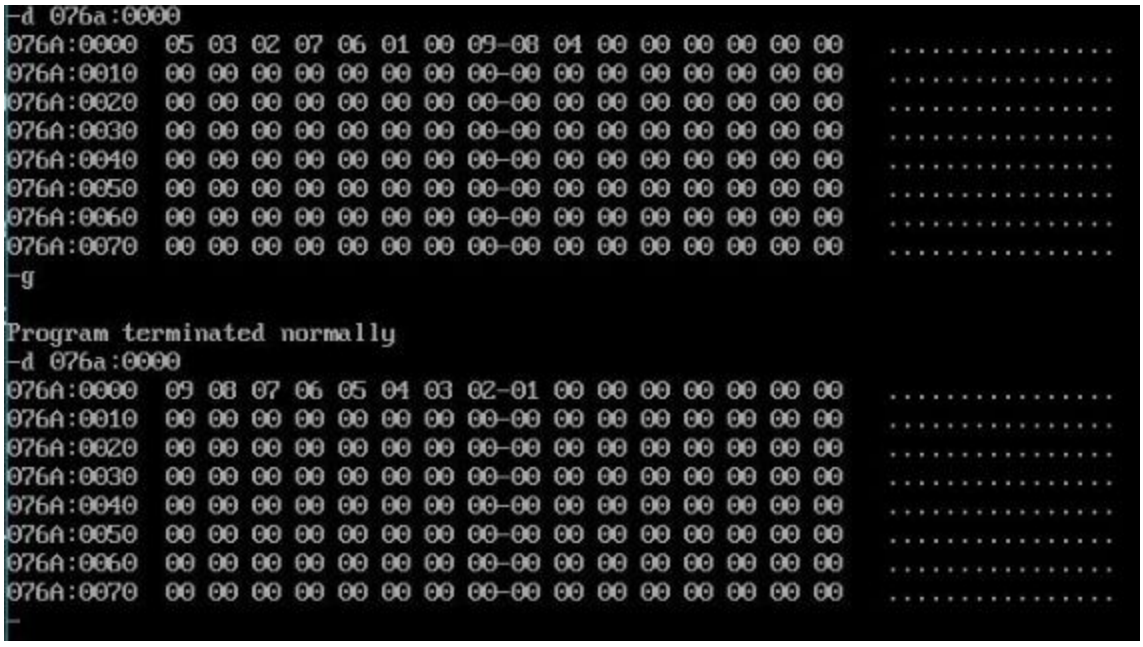
Program	Comments
assume cs:code,ds:data	Data and code segments are initialized
data segment array db 05h,03h,02h,07h,06h,01h,00h,09h,08h,04h data ends	
code segment org 0100h	Code segment begins Originating address is set to 0100h
start : mov ax,data mov ds,ax mov ch , 09h	AX ←data DS ← AX CH ← 09h (outer loop count)
loop1 : mov si ,offset array mov cl , 09h	Move the offset of array to SI CL ← 09h

loop2 : mov al , [si]
mov ah , [si+1]
cmp ah , al
jc skip
xchg al,ah
mov [si] ,al
mov [si+1],ah
 	
skip: inc si
dec cl
jnz loop2
dec ch
jnz loop1
 	
mov ah,4ch
int 21h
code ends
end start
	Move [SI] to AL and [SI+1] to AH registers
	
	Compare AH and AL
	If there is a carry(i.e they are in correct order) jump to skip else swap AL and AH
	Move AL to [SI] and AH to [SI+1] again
	
	SI ← SI + 1
	CL ← CL - 1
	if CL is not zero jump to loop2.
	CH ← CH - 1
	if CH is not zero jump to loop1
	
	Program terminates

Unassembled code:



Sample Input and Output:



Result:

Thus, assembly language programs for sorting in ascending and descending order have been executed and the output has been verified.

Exp no: 7

BCD Addition and Subtraction

Aim:

To execute assembly language programs for performing BCD addition and subtraction of two 8-bit numbers using MASM software.

BCD Addition

Algorithm:

1. Start
2. Use the define byte(db) directive to declare byte type variables- opr1, opr2, result
3. and carry. Initialise opr1, opr2 with some hex values and result and carry with 0 in hexadecimal.
4. Load data from the data segment into the DS register
5. Transfer data from opr1, opr2 to AL, BL respectively.
6. Transfer the value 00H into the CH register to keep track of the carry
7. Add the contents of registers AL, BL
8. Decimal adjust the result stored in AL to get the correct BCD result using DAA instruction.
9. Jump to step 11 if there is no carry
10. Increment carry in the CH register
11. Store the result of the DAA operation in result and move the contents of CH register to carry
12. Terminate the program

Program:

Program	Comments
assume cs:code,ds:data data segment	Declare the segments for CS and DS register

opr1 db 11h	Var opr1 ← 11h
opr2 db 99h	Var opr2 ← 99h
result db 00h	Var result ← 00h
carry db 00h	Var carry ← 00h
data ends	
code segment	
org 0100h	Specify the start of the code segment
start: mov ax,data	AX ← data
mov ds,ax	DS ← AX
mov al , opr1	AL ← opr1
mov bl , opr2	BL ← opr2
mov ch , 00h	CH ← 00h
add al , bl	AL ← AL + BL
daa	Adds 6 to the digit>9, if both the digits are >9 then 66h is added.
jnc here	If no carry is generated, jump to the label “here:”
inc ch	If jump not executed, increment carry
here: mov result,al	result ← AL
mov carry ,ch	carry ← CH
mov ah,4ch	AH ← 4ch
int 21h	Interrupt causes the program to exit if AH has a value of 4ch
code ends	
end start	

Unassembled code:

```
D:\>debug 7a.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 B500          MOV     CH,00
076B:010E 02C3          ADD     AL,BL
076B:0110 27            DAA
076B:0111 7302          JNB     0115
076B:0113 FEC5          INC     CH
076B:0115 A20200        MOV     [0002],AL
076B:0118 882E0300      MOV     [0003],CH
076B:011C B44C          MOV     AH,4C
076B:011E CD21          INT     21
-_
```

Sample Input and Output:

```
-g

Program terminated normally
-d 076a:0000
076A:0000  11 99 10 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   ................
-_
```

```
D:\>debug 7a.exe
-e 076a:0000
076A:0000  11.38   99.45

-g

Program terminated normally
-d 076a:0000
076A:0000  38 45 83 00 00 00 00 00-00 00 00 00 00 00 00 00   8E..............
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   ................
```

BCD Subtraction

Algorithm:

1. Start
2. Use the define byte(db) directive to declare byte type variables- opr1, opr2, result
3. and carry. Initialise opr1, opr2 with some hex values and result and carry with 0 in hexadecimal.
4. Load data from the data segment into the DS register
5. Transfer data from opr1, opr2 to AL, BL respectively.
6. Transfer the value 00H into the CH register to keep track of the carry
7. Subtract the contents of registers AL, BL
8. Decimal adjust the result stored in AL to get the correct BCD result using DAS instruction.
9. Jump to step 13 if there is no carry
10. Increment carry in the CH register
11. Find the 10's complement of the result by : 99 - result + 01
12. Decimal adjust the 10's complement
13. Store the result of the DAS operation in result and move the contents of CH register to carry
14. Terminate the program

Program:

Program	Comments
assume cs:code,ds:data	Declare the segments for CS and DS register
data segment	
opr1 db 33h	Var opr1 ← 33h
opr2 db 35h	Var opr2 ← 35h
result db 00h	Var result ← 00h
carry db 00h	Var carry ← 00h
data ends	
code segment	
org 0100h	Specify the start of the code segment
start: mov ax,data	AX ← data
mov ds,ax	DS ← AX
mov al , opr1	AL ← opr1
mov bl , opr2	BL ← opr2
mov ch , 00h	CH ← 00h
sub al , bl	AL ← AL-BL
das	Decimal adjust AL after subtraction, i.e subtract 6 from the digit > 9
jnc here	If no carry is generated, jump to the label “here:”
inc ch	If jump not executed, increment carry
mov cl, 99h	CL ← 99h
sub cl,al	CL ← CL - AL
add cl ,ch	CL ← CL + 01
mov al,cl	AL ← CL
das	Decimal adjust AL
here: mov result,al	result ← AL
mov carry ,ch	carry ← CH
mov ah,4ch	AH ← 4ch
int 21h	Interrupt causes the program to exit if AH has a value of 4ch
code ends	
end start	

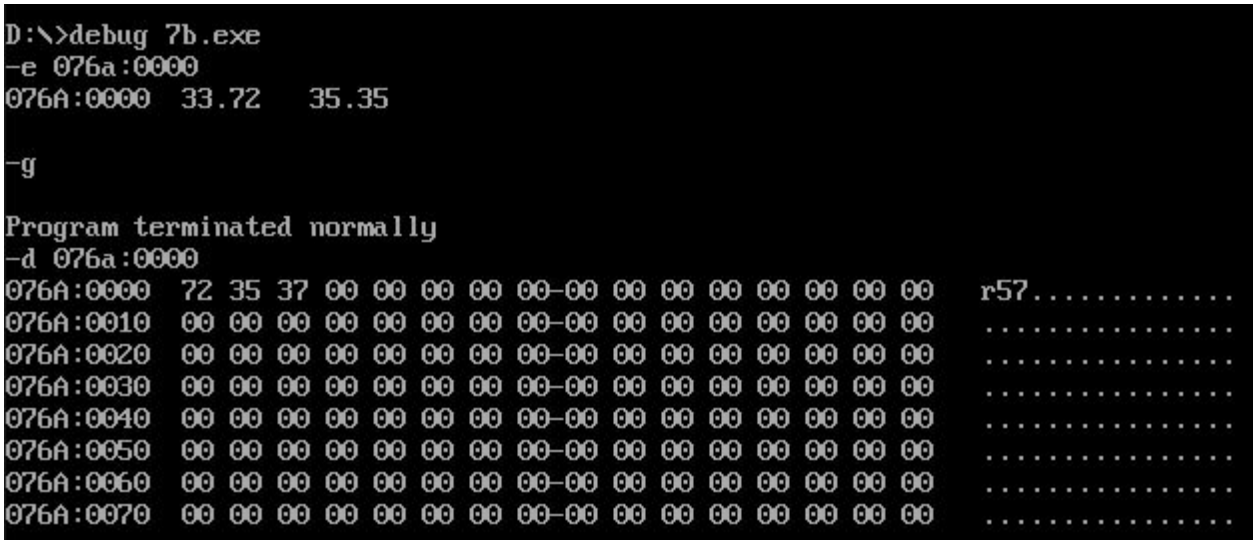
Unassembled code:

```
D:\>debug 7b.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 B500          MOV     CH,00
076B:010E 2AC3          SUB     AL,BL
076B:0110 2F            DAS
076B:0111 730B          JNB     011E
076B:0113 FEC5          INC     CH
076B:0115 B199          MOV     CL,99
076B:0117 2AC8          SUB     CL,AL
076B:0119 02CD          ADD     CL,CH
076B:011B 8AC1          MOV     AL,CL
076B:011D 2F            DAS
076B:011E A20200        MOV     [0002],AL
```

Sample Input and Output:

```
-g

Program terminated normally
-d 076a:0000
076A:0000  33 35 02 01 00 00 00 00-00 00 00 00 00 00 00 00   35..............
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:

Thus, assembly language programs for 8-bit BCD addition and subtraction have been executed and the output has been verified with different values for these operations.